Universidad Carlos III de Madrid

Departamento de Estadística

PROYECTO FIN DE CARRERA

INFRAESTRUCTURA WEB PARA LA IMPLEMENTACIÓN DE ESTRATEGIAS DE INVERSIÓN AUTOMÁTICAS DE BAJA VOLATILIDAD

Autor: Pablo Pérez González

Tutores: Francisco Javier Nogales y Alberto Martín Utrera

Leganés, diciembre de 2013

Título: Infraestructura web para la implementación de estrategias de inversión automáticas de baja volatilidad.
Autor: Pablo Pérez González
Directores: Javier Nogales y Alberto Martín Utrera

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Muchos años han pasado desde que empecé la carrera y muchas son las personas que he conocido y me gustaría citar. No sólo profesores y compañeros sino también familia y amigos. Me gustaría poder nombrarlos a todos pero me temo que no sería posible.
Gracias a mis dos profesores del proyecto por la confianza depositada Javier y Alberto así como la ayuda suministrada a lo largo del proyecto.
Gracias a mi familia, a mis padres y mi hermana, por compartir mis alegrías y mis sufrimientos por todo ese apoyo moral en los peores momentos y el infatigable ánimo que sólo los seres más cercanos de tu vida te pueden dar.
Gracias también a todos mis compañeros de carrera, en especial a Héctor por esos buenos momentos finales en los que la solidaridad y el entendimiento mutuo fueron de vital ayuda.
Pero sobre todo quiero agradecer a una persona en especial el haber acabado esta etapa, sin su apoyo, su entereza o sus consejos esta carrera habría sido mucho más difícil, más tediosa y vacía. Gracias Lucía.

Gracias.

Resumen

Los sistemas automáticos de trading son programas automáticos que realizan operaciones de compra venta de activos negociables. En este proyecto se tratará de generar una infraestructura web que presente de forma automática y clara una serie de carteras de inversión. Las carteras ofrecidas estarán compuestas de una serie de activos elegidos entre dos índices de inversión. El criterio de selección trata de optimizar la volatilidad de nuestra inversión de manera que sea inferior a los índices elegidos mientras mantenemos unas rentabilidades similares. Las rutinas de selección han sido suministradas por Javier Nogales y Alberto Martín.

Se presentarán en la web un total de 8 carteras de inversión. Operarán sobre las empresas de los índices Eurostoxx 50 para Europa y S&P 500 en el mercado americano. Las carteras se dividirán según el mercado y dependiendo del número de activos que la componen, cartera grande o pequeña. Cómo elemento diferenciador frente a otros sistemas se ofrece además para cada mercado una recomendación distinta para cada divisa, de manera que se pueda invertir tanto en Europa como en Norteamérica en la divisa deseada, euros o dólares.

Palabras clave: Carteras de Inversión, baja volatilidad, procesado de datos, página web, ganancia, ratio de Sharpe.

Abstract

An automated trading system is a computer trading program that automatically performs trades to an exchange. In this project we will try to create a structure that presents automatically and concise several investment portfolios. The portfolios offered will consist of a number of assets chosen between two different indexes. The selection criteria is to optimize the volatility of our investment while maintaining returns. The selection routines have been supplied by Javier Nogales and Alberto Martín.

It will be presented 8 investment portfolios on the web. We will be operating on the companies from Eurostoxx 50 for Europe and S&P 500 in the U.S. market. Portfolios are divided according to the market and depending on the number of assets composing it, large and small portfolios. In order to be different from other systems it is also offered a different recommendation regarding the currency for each market, so that you can invest in both, Europe and North America in the desired currency, Euros or Dollars.

Keywords: Investment portfolios, low volatility, data processing, web page, income, profits, Sharpe ratio.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

Este primer capítulo servirá para explicar y definir el proyecto. Se explicarán brevemente los conceptos de servidores, mercados de valores e inversiones de bajo riesgo. Se expondrán además los objetivos que se pretenden alcanzar, los medios utilizados para ello y la estructura de la memoria.

1.1 Introducción

El objetivo principal de este proyecto es la creación de una infraestructura web que presente a posibles usuarios interesados una serie de carteras de inversión.
Las empresas elegidas serán escogidas entre los índices S&P 500 y Eurostoxx 50. Las estrategias que utilizaremos para seleccionar las empresas donde invertiremos tratarán de minimizar la volatilidad a la vez que se mantienen los retornos de los índices seleccionados. Estas estrategias han sido suministradas por los tutores del proyecto y no han sido desarrolladas en este proyecto. Las tareas que si han sido llevadas acabó en el presente proyecto son las siguientes.

- Descarga de datos históricos de servidores externos. Incluye diferentes tipos de datos, tickers, cambios de moneda o valores de cotización.
- Tratamiento inicial de dichos datos para su adecuación a las rutinas suministradas.

- Uso de las rutinas suministradas para calcular históricos. Ajustando parámetros para mejorar el rendimiento.
- Tratamiento posterior de los resultados obtenidos. Cálculo de medidas de rendimiento y adecuación a la presentación web.
- El total de la infraestructura web. Desde la parte lógica a la visual.

Los valores que utilizaremos serán datos ofrecidos por terceros de manera gratuita, el caso más crítico son los valores de cotización que se obtendrán a partir de Yahoo Finance. A lo largo del proyecto se ha intentado minimizar en la medida de lo posible errores y fallos que pudieran contener dichos datos, adecuando las matrices obtenidas a las rutinas suministradas por los tutores del proyecto.

Para generar un primer volumen de datos y poder ofrecer rendimientos históricos que atraigan a inversores, se realizará una primera simulación histórica, cómo si el sistema hubiese estado en marcha 5 años. A partir de esos datos se actualizarán todas las medidas y gráficos semanalmente. Esto se consigue ejecutando unas rutinas en el servidor que actualizan los datos ofrecidos en la web cada semana, aunque sería posible variar la frecuencia de actualización con facilidad.

Los datos presentados en la web han de ser accesibles al mayor rango de usuarios posibles, por lo que será importante mantener en la web una presentación sencilla y ordenada de las medidas que estamos ofreciendo.

Debido a la naturaleza de este proyecto se necesita aplicar diversos conceptos pertenecientes a distintas disciplinas, relacionadas algunas débilmente entre sí. A grandes rasgos tendremos una parte ligada a la informática, sobre todo en la parte del servidor, otra ligada a conceptos de mercado y otra referente a la estadística.
Para facilitar la comprensión de todo lo realizado pasaremos a definir diferentes conceptos relacionados y utilizados en el proyecto.

Servidores

Estrictamente un servidor es un tipo de software que realiza tareas a petición de distintos usuarios. Este término también se utiliza para referirse al ordenador físico donde ejecutaremos el software. Este hardware o máquina tiene como función proveer determinados datos para que otras máquinas o personas puedan utilizarlos. Un ejemplo de este doble significado podemos encontrarlo si por ejemplo hablamos de un servidor web. Podemos referirnos tanto a la máquina que almacena y gestiona las páginas o aplicaciones web cómo al servidor http que encontramos en su interior, es decir, al software Apache por ejemplo. Este software es el encargado de entregar y manejar los componentes de las páginas web cuando alguien realiza una petición como podría ser el caso de un navegador[1].

Dentro de los servidores de internet podemos encontrar de muchos tipos, que realizan diversas funciones, sin embargo todos tienen en común de proporcionar acceso a archivos o servicios. Dentro de los diferentes tipos de servidores podemos nombrar algunos:

- Servidor de Archivos: almacena y sirve ficheros.

- Servidor de Correo: se encarga de gestionar los flujos de correos electrónicos entre usuarios.
- Servidor Proxy: servidor puente que puede almacenar en una caché temporal ficheros, o páginas web.
- Servidor DNS: permite establecer una conexión a una determinada IP usando únicamente un nombre de dominio.
- Servidor Web: almacena contenido web y lo distribuye a los usuarios que lo solicitan.

El servidor que usaremos en este proyecto será un servidor web que mediante el protocolo http ofrecerá una página web con contenido dinámico. Los usuarios de este servidor podrán acceder a los datos de este servidor mediante un navegador web. Indirectamente también existirá un servidor DNS que asociará un nombre de dominio con la dirección IP de nuestro servidor.

Mercados de valores

Un mercado de valores es un tipo de mercado financiero donde se ofrecen y demandan fondos de renta variable y renta fija de una manera estructurada. La compraventa se realiza a partir de valores negociables, bonos, acciones, obligaciones... Comúnmente nos referimos a estos mercados como Bolsa, en las bolsas de valores existen un conjunto de instituciones y agentes financieros que negocian los distintos tipos de activos.
El objetivo principal de estos mercados es la captación de capital extra que ayude en mayor o menos medida a la financiación de la empresa. A veces esta compraventa conlleva especulación por parte de las empresas o inversores. Los intercambios en estos mercados son a priori públicos y transparentes, aunque también existen colocaciones privadas[2].

En este sistema entraremos en la compraventa en tiempo real de valores ya emitidos y vendidos en el mercado que circulan entre los diferentes inversores. Dentro de los mercados podemos encontrar índices de valores, que constituyen conjuntos de acciones elegidas con cierto criterio, como puede ser el volumen de las empresas o divididas en un determinado sector[3]. Las compañías de los índices que manejaremos serán las más representativas en sus países y estarán sujetas a sus cambios y ciclos económicos. Los dos índices elegidos son el Eurostoxx 50 y el S&P 500.

Estrategias de inversión automáticas

Los sistemas automáticos de trading son generalmente programas informáticos creados e instalados en computadoras que ejecutan órdenes de compra/venta de activos negociables. Las órdenes de compra/venta no tienen porque ser realizadas por el ordenador, también pueden ser efectuadas por una persona, siendo el programa el encargado de avisar si es necesario realizar una nueva operación.

Si bien estos sistemas son relativamente nuevos y desconocidos por algunos inversores, en este proyecto ofreceremos una estrategia de inversión automática definida y facilitaremos las carteras y datos a posibles inversores interesados, de una forma sencilla y sin necesidad de entrar en detalles de programación.

Inversiones de baja volatilidad

Las inversiones de baja volatilidad son una estrategia de inversión que intentan minimizar los cambios bruscos en las ganancias obtenidas. La volatilidad es el riesgo que corre un activo, y debe entenderse como la fluctuación que puede sufrir un activo en el tiempo. La volatilidad mide la frecuencia e intensidad de los cambios en el precio de un activo[4].
Suelen ser consideradas estrategias conservadoras, pero no necesariamente de menor ganancia. Suelen basarse en la diversificación del capital en distintos valores no dependiendo únicamente del comportamiento de una sola empresa. El principal objetivo de estas inversiones es reducir el riesgo de las inversiones manteniendo o mejorando la rentabilidad[5].
Estas inversiones tienen también un carácter a largo plazo, quizá a corto plazo no se obtengan tan buenos resultados como con otras estrategias, pero con la inversión de valores de baja volatilidad puedes generar buenos resultados finales sin necesidad de obtener ganancias a corto plazo excepcionales.
Es un concepto que ya existía desde hace muchos años sin embargo está siendo utilizado de forma más habitual en la actualidad, motivado en gran parte por la gran crisis financiera del 2007-2008.

1.2 Objetivos

El objetivo fundamental de la tesis es la creación de un sistema web para la realización de estrategias de inversión automáticas basadas en técnicas cuantitativas, con buen comportamiento en términos de rentabilidad-riesgo. Debe ser útil para pequeños y grandes inversores. A lo largo del desarrollo se utilizarán conocimientos y aptitudes adquiridas durante la carrera
La funcionalidad básica a implementar es:

- Descarga y preparado de los datos de entrada a unas estrategias suministradas.
- Análisis de los datos de salida de las estrategias.
- Ajuste de parámetros óptimos y selección de estrategia que se ajuste mejor a los criterios nombrados.
- Generación de los datos a representar en la página web.
- Página web con las carteras de recomendación.

1.3 Medios empleados

Necesitaremos una serie de medios para poder llevar a cabo el desarrollo de este proyecto. El sistema integra diferentes tecnologías y varios lenguajes de programación. Para poder trabajar con estos elementos han sido necesarios diversos programas y entornos de desarrollo. Se nombrarán tanto herramientas de software como de hardware.

Software

La lista completa de los lenguajes utilizados es la siguiente:

- Java
- HTML
- CSS
- Apache Tomcat
- Matlab

Los entornos de desarrollo y programas de edición utilizado con su correspondiente versión y fabricante son los mostrados en la Tabla 1.

Programa	Desarrollador	Versión
NetBeans IDE	Sun Microsystems/Oracle Corporation	7.3
Matlab	The MathWorks	R2010a y R2012a
Notepad++	Notepad++ team	6.5.2
ProjectGA	Equipo de GanttProject 2012	2.6.4

Tabla 1. Entornos de desarrollo y programas.

Hardware

Los dispositivos de la Tabla 2 fueron utilizados para la realización del proyecto.

Dispositivo	Características	Función
Portátil	• Intel Core i7 2.66GHz • 8 GB RAM • OS Windows 7 Professional	• Desarrollo de proyecto • Servidor de pruebas
PC de sobremesa	• Intel Core i7 2.80 GHz • 8 GB RAM • OS Windows Server 2012	• Servidor real remoto

Tabla 2. Dispositivos utilizados.

1.4 Estructura de la memoria

Resumiremos brevemente los capítulos en los que se compone este proyecto, para facilitar la lectura y comprensión del proyecto.

Capítulo 1. Capítulo de presentación donde se explican los conceptos utilizados en el proyecto la motivación y los recursos utilizados.

Capítulo2. Descripción de las tecnologías y herramientas utilizadas en la creación de este proyecto.

Capítulo 3. Desarrollo técnico y detallado de toda la arquitectura del sistema. Se presentan y explican todos los módulos involucrados.

Capítulo 4. Explicación sobre las distintas fases de las que consta el proyecto. Y desglose de todos los costes que conlleva su implementación.

Capítulo 5. Se expondrán las diferentes conclusiones derivadas tanto del desarrollo cómo de los resultados del proyecto. Además se ofrecerán posibles líneas de desarrollo futuras.

Capítulo 6. Glosario de los acrónimos utilizados en el proyecto.

Capítulo 7. Bibliografía con la documentación consultada para el desarrollo del proyecto.

Capítulo 2

Tecnologías y plataformas

En este capítulo se repasarán cada una de las tecnologías utilizadas en este proyecto de fin de carrera. Para poder integrar obtener y mostrar todos los datos de las carteras de inversión han sido necesarias diversas tecnologías. Esta capítulo no profundizará en cómo se han utilizado estas tecnologías solo se nombrarán y se dará una visión general.
Se dará una breve descripción sobre su funcionamiento, exponiendo primero los lenguajes de programación utilizados y a continuación se expondrá el entorno de desarrollo empleado. Finalmente se explicarán los dos índices de cotización utilizados para los cálculos.

2.1. Matlab

Matlab no es sólo un entorno de desarrollo interactivo, también es considerado un lenguaje de alto nivel para desarrollo, visualización y programación. Esta herramienta nos proporciona funciones y librerías con las que analizar datos, desarrollar algoritmos, crear modelos y aplicaciones. Este lenguaje está optimizado para operar con matrices y alcanzar resultados más rápido que otros lenguajes tradicionales de programación[6].

Durante mucho tiempo hubo críticas porque MATLAB es un producto propietario de The Mathworks, y los usuarios están sujetos y bloqueados al vendedor. Recientemente se ha proporcionado una herramienta adicional llamada MATLAB Builder bajo la sección de herramientas "Application Deployment" para utilizar funciones MATLAB como archivos de biblioteca que pueden ser usados con ambientes de construcción de aplicación .NET o Java. Pero la desventaja es que el computador donde la aplicación tiene que ser utilizada necesita MCR(MATLAB Component Runtime) para que los archivos MATLAB funcionen correctamente. MCR se puede distribuir libremente con los archivos de biblioteca generados por el compilador MATLAB[7].

2.2 Servidor

Página web

Una página web es un documento de información electrónica que puede contener diverso contenido, ya sea en formato de sonido, video, texto, enlaces u otras formas. Esta información esta codificada generalmente en html o xhtml, en nuestro caso xhtml. A partir de estas páginas se proporciona navegación mediante enlaces a otras páginas web.

Las páginas estarán almacenadas en un equipo remoto ubicado en la Universidad Carlos III de Madrid y podremos acceder a ellas usando el protocolo de transferencia de hipertexto (http).

Las páginas web podrán contener información estática o dinámica. La información dinámica requiere de lógica adicional que el lenguaje xhtml no provee, por ejemplo se pueden utilizar los lenguajes javascript o php.

Una página web es en esencia una tarjeta de presentación digital, ya sea para empresas, organizaciones, o personas, así como una tarjeta de presentación de ideas y de informaciones y de teorías. Así mismo, la nueva tendencia orienta a que las páginas web no sean sólo atractivas para los internautas, sino también optimizadas (preparadas) para los buscadores a través del código fuente[8].

Apache Tomcat

Apache Tomcat es una implementación software de código abierto para servlet Java y tecnologías JSP. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAs. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de

transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad. Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la maquina virtual Java[9].

HTML

HTML hace referencia al lenguaje de marcado predominante para la elaboración de páginas web. Básicamente se trata de un conjunto de etiquetas que sirve para definir el texto y otros elementos que compondrán una página web. Se utiliza para describir y traducir la estructura y la información en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de etiquetas, rodeadas por corchetes angulares (<,>), en un documento de texto[10].

También puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir o hacer referencia a un tipo de programa llamado script, el cual puede afectar al comportamiento de navegadores web y otros procesadores de HTML.

Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria[11].

Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

Java EE

Java Platform, Enterprise Edition o Java EE es una plataforma de programación—parte de la Plataforma Java—para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java. Permite utilizar arquitecturas de N capas distribuidas y se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

La plataforma Java EE está definida por una especificación. Java EE es también considerada informalmente como un estándar debido a que los proveedores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE.

Java EE tiene varias especificaciones de API, tales como JDBC, RMI, e-mail, JMS, Servicios web, XML, etc y define cómo coordinarlos. Java EE también configura algunas especificaciones únicas para Java EE para componentes. Estas incluyen Enterprise JavaBeans, servlets, portlets(siguiendo la especificación de

Portlets Java), JSP y varias tecnologías de servicios web. Ello permite al desarrollador crear una Aplicación de Empresa portable entre plataformas y escalable, a la vez que integrable con tecnologías anteriores. Otros beneficios añadidos son, por ejemplo, que el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de en tareas de mantenimiento de bajo nivel.

JSF

JavaServer Faces (JSF) es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE [12].
JSF incluye:

- Un conjunto de APIs para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entrada, definir un esquema de navegación de las páginas y dar soporte para internacionalización y accesibilidad.
- Un conjunto por defecto de componentes para la interfaz de usuario.
- Dos bibliotecas de etiquetas personalizadas para JavaServer Pages que permiten expresar una interfaz JavaServer Faces dentro de una página JSP.
- Un modelo de eventos en el lado del servidor.
- Administración de estados.
- Beans administrados.

CSS

CSS es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura.
CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los Estilos definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores web controlar el estilo y el formato de múltiples páginas web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento.

El funcionamiento de CSS se basa en reglas, es decir, declaraciones sobre el estilo de uno o más elementos. Las hojas de estilo están compuestas por una o más de esas reglas aplicadas a un documento HTML o XML. La regla tiene dos partes: un selector y la declaración. A su vez la declaración está compuesta por una propiedad y el valor que se le asigne[13].

2.3 Entorno de desarrollo

En este apartado se explica brevemente que entorno de desarrollo se ha decidido utilizar y porque. Si bien Matlab también es un entorno desarrollo se ha colocado en un apartado diferente debido a que además es un lenguaje de alto nivel.

NetBeans IDE

NetBeans es un entorno de desarrollo integrado, pensado en un principio para ser usado con Java sin embargo gracias a la inclusión de diferentes módulos dispone de diferentes herramientas que extienden su funcionalidad. Además es gratuito y sin restricciones de uso[14].

Esta plataforma permite desarrollar aplicaciones a partir de un conjunto de módulos. Esto permite ampliar aplicaciones creadas agregándole nuevos módulos.
Al ser una página web implementada en java la utilización de NetBeans resulta idónea para este caso, además facilita la integración de JSP y Apache Tomcat incluyendo además por defecto las librerías de Primefaces.

2.4 Librerías externas

En este apartado se exponen las diferentes librerías utilizadas Primefaces 3.5 y javacsv para la página web en NetBeans y CVX en el entorno Matlab.

Primefaces

Primefaces es una librería de libre distribución para JSF con varias cualidades, entre ellas:

- Un amplio grupo de Componentes: gráficos diálogos, tablas...
- Ajax
- Distribuido en un único fichero .jar
- Amplia comunidad de usuarios
- Documentación extensa

Provee una serie de componentes con una elegante estética y ampliada funcionalidad, siendo de gran ayuda para presentar servicios web de manera atractiva a posibles clientes[15].

Javacsv

Java CSV es una librería pequeña y rápida de libre distribución que nos permite leer y escribir archivos csv y otros ficheros de texto delimitados (por comas, puntos,

etc...)[16]. Esta herramienta permite al servidor leer los archivos generados por Matlab para su posterior tratamiento y muestra en la página web.

CVX

Este módulo fue añadido a Matlab cómo ayuda a la hora de aplicar las diferentes estrategias de inversión. CVX es un sistema de optimización convexa[17] basado en Matlab[18]. Dado que nuestras funciones de decisión necesitan de la resolución de funciones convexas utilizamos esta herramienta para acelerar y optimizar los cálculos a la hora de calcular los pesos de nuestro portfolio.

2.5 Índices de referencia

A lo largo de todo el proyecto se va a trabajar únicamente con dos mercados. Uno europeo Eurostoxx 50 y otro americano S&P 500. Hay que tener en cuenta que cada uno opera en su propia moneda euros y dólares respectivamente

Eurostoxx 50

El índice Eurostoxx 50 agrupa las 50 empresas más importantes de la zona euro. La variable que se usa para decidir que empresas entran es la mayor ponderación por capitalización bursátil.

Por países, alrededor del 36 por ciento del total son francesas, por detrás se sitúan las alemanas 32 por ciento y las españolas 12 por ciento[19].

S&P 500

El índice Standard and Poor’s 500 (S&P 500) es uno de los índices de referencia más importantes en Estados Unidos. Se trata de un índice formado por las cotizaciones de las 500 empresas más importantes que cotizan en la Bolsa de Nueva York. Su nombre deriva de la empresa de rating “Standard and Poor’s”, que fue la creadora del mismo.

Se diferencia de otros índices bursátiles de Estados Unidos, como el Dow Jones Industrial Average y el Nasdaq, debido a su electorado diverso y metodología de ponderación. Este índice se pondera de acuerdo a la capitalización de mercado de cada una de las empresas[20].

Capítulo 3

Análisis del sistema

En este capítulo se explicará cómo funciona en detalle el sistema completo. A diferencia del capítulo anterior sí se entrará en profundidad sobre detalles técnicos del sistema. Dividiremos el capítulo en 3 grandes apartados (Figura 1). El primero englobará todo el procesamiento realizado mediante el programa Matlab. El segundo apartado mostrará cómo se generan los ficheros con los datos de rendimiento de las estrategias de inversión, que posteriormente serán utilizadas por la plataforma web, los ficheros serán generados también con Matlab. Por último el tercer apartado explicará cómo ha sido desarrollada la página web.

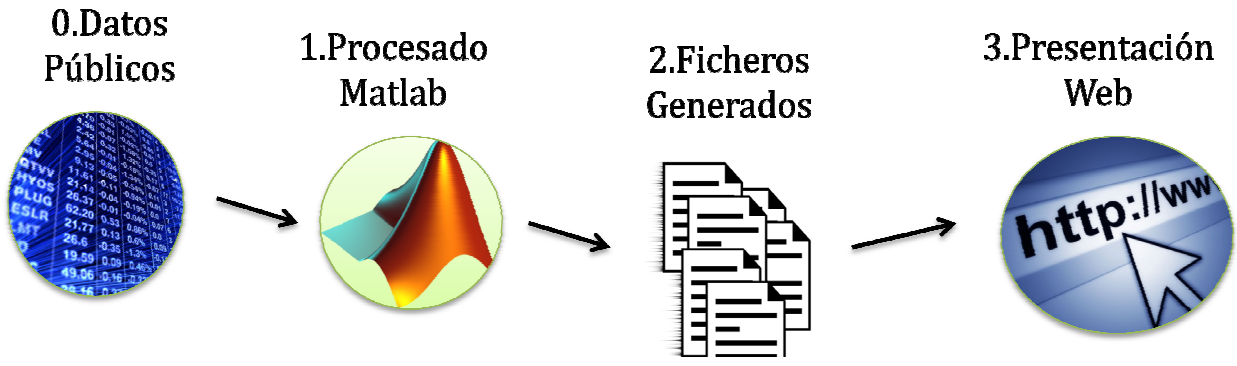

Figura 1. Estructura general.

3.1 Procesado Matlab

Dentro del procesado de Matlab son necesarias unas aclaraciones. Al operar en dos mercados con diferente divisa y de distinto volumen de empresas será necesario tener una rutina diferente para cada uno de ellos, por tanto para la generación y actualización de los ficheros de datos, usaremos 4 rutinas en total:

- Simulación histórica Eurostoxx (startOneEU.m)
- Simulación histórica S&P 500 (startOneSP.m)
- Rutina periódica de actualización Eurostoxx (weeklyEU.m)
- Rutina periódica de actualización S&P 500 (weeklySP.m)

Algunas de las diferencias entre mercados de estas rutinas serán entre ellas, el diferente ajuste en determinados parámetros o la descarga de tickers según el mercado. A lo largo de este apartado explicaremos las dos rutinas principales simulación histórica y actualización de forma genérica distinguiendo mercados únicamente en los puntos donde sea necesario.

Las rutinas de simulación histórica de manera genérica tendrán la siguiente función, descarga de datos suficiente como para generar 5 años de datos, se simulará la rentabilidad de nuestro sistema y otros parámetros de rendimiento así como las carteras calculadas y sus pesos a lo largo de todo el periodo cómo si el sistema hubiese estado en marcha estos 5 años con los datos ofrecidos por Yahoo. Además se ofrecerá una comparación con los datos de los índices del Eurostoxx y S&P 500 también suministrados por Yahoo.

Para usar estas rutinas tanto simulación histórica cómo actualización semanal ha sido necesario unos ajustes previos, entre ellos hemos tenido que seleccionar la estrategia óptima de inversión entre 3 distintas disponibles, y se han ajustado también parámetros relacionados con la inversión.

Las rutinas de actualización se encargarán de poner al día semana a semana los datos ofrecidos en la web aplicando rebalanceo de carteras cuando sea necesario o simplemente evolucionando los pesos de la cartera actual en caso contrario.

Estructuraremos este apartado de la siguiente manera, rutina de simulación histórica, rutina de actualización semanal y rutinas de ajuste de parámetros y estrategias. En el servidor únicamente ejecutaremos la rutina semanal. Las rutinas de ajuste o de simulación histórica hacia atrás han sido ejecutadas de forma aislada en otro sistema.

3.1.1 Simulación histórica.

Para poder ofrecer los primeros datos a los usuarios será necesario hacer una simulación con datos históricos para la puesta en marcha del servidor. En la página web mostraremos varias medidas y gráficos enseñando la evolución de nuestras

estrategias a lo largo de 5 años. Esta rutina será la encargada de obtener y calcular todos los datos necesarios para esta primera inicialización.

Las rutinas que usaremos serán *startOneEU.m* y *startOneSP.m*, dentro de estas rutinas también se calcularán todos los datos y gráficas a representar, si bien serán explicados en el apartado 3.2. Cada rutina generará su propia carpeta de datos que posteriormente serán utilizados por el servidor para generar la página web, ver Figura 2.

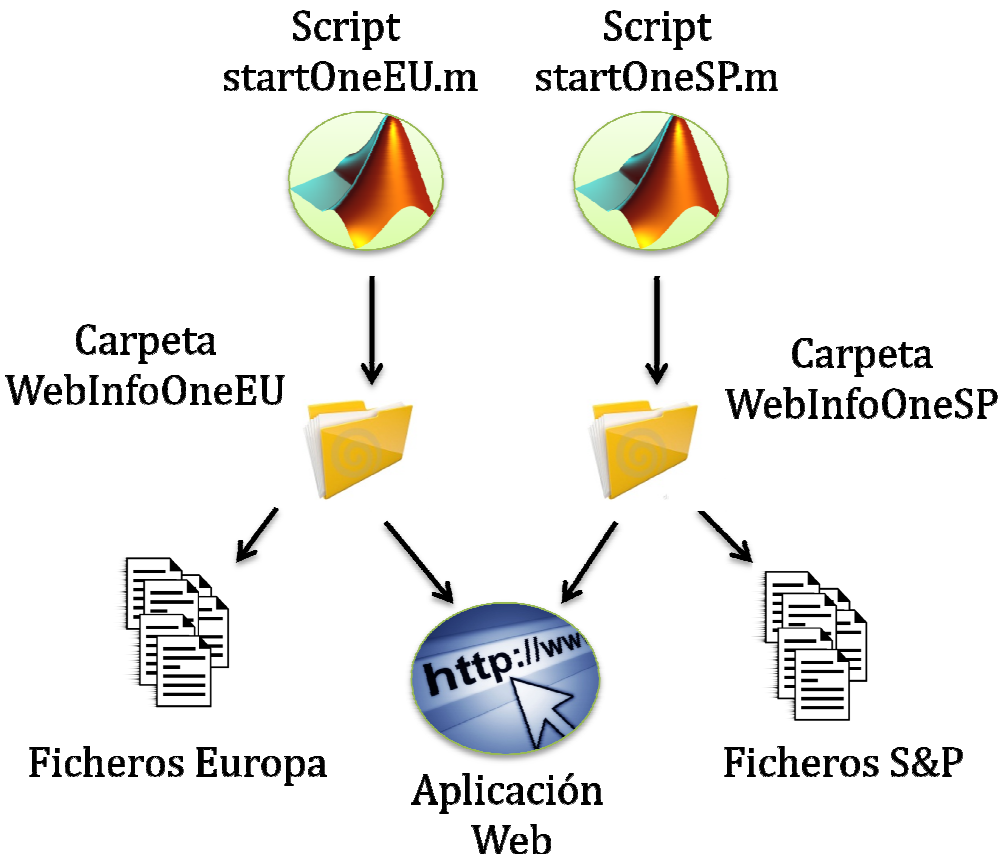

Figura 2. Esquema rutinas históricas.

Ambas estrategias ofrecerán además 4 carteras distintas para cada mercado. Una pequeña y una grande y a su vez estás ofrecerán una en euros y otra en dólares. En total ofreceremos 8 carteras, 4 para Eurostoxx y 4 para S&P 500:

Ficheros Europa: startOneEU.m

- Cartera grande Eurostoxx en euros.
- Cartera grande Eurostoxx en dólares.
- Cartera pequeña Eurostoxx en euros.
- Cartera pequeña Eurostoxx en dólares.

Ficheros S&P 500: startOneSP.m

- Cartera grande S&P 500 en dólares.
- Cartera grande S&P 500 en euros.
- Cartera pequeña S&P 500en dólares.
- Cartera pequeña S&P 500 en euros.

Las operaciones realizadas por esta simulación histórica y que serán detalladas a continuación son las mostradas en la siguiente figura ver Figura 3.

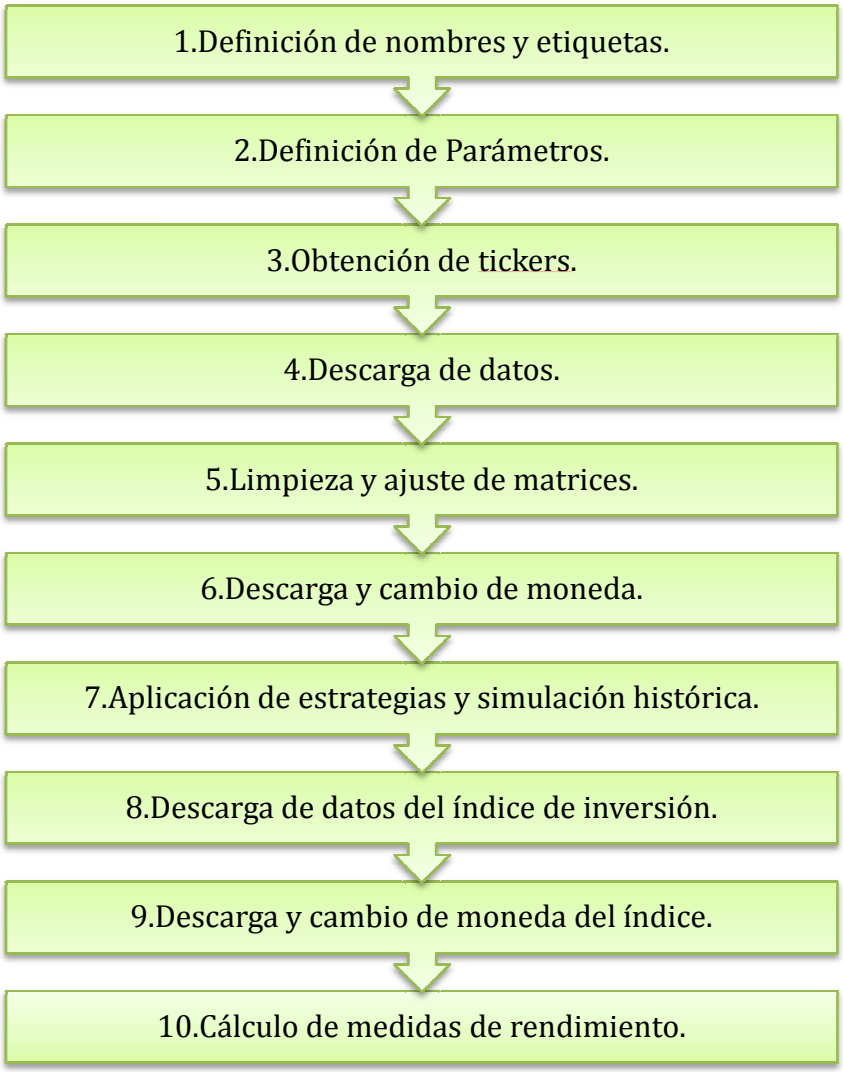

Figura 3. Operaciones rutinas históricas.

1. Definición de nombres y etiquetas.

Al inicio de la simulación histórica definiremos todos los nombres de los ficheros que posteriormente generaremos así como las etiquetas de las gráficas, título de la gráfica, nombres en los ejes y leyendas. También definiremos la ruta donde

colocaremos los archivos. Cada mercado dispondrá de su propio directorio donde escribirá los ficheros resultantes, carpetas webInfoOneEU y webInfoOneSP.

2. Definición de parámetros.

En este paso definimos los parámetros que serán necesarios a la hora de simular las estrategias, nos servirán para optimizar el funcionamiento de nuestro sistema.

Destacaremos los más importantes y su finalidad, entre paréntesis está escrito el nombre exacto de la variable usada en Matlab:

- Periodo de rebalanceo ("rebalance"): será el intervalo de días que pasará entre reajuste de carteras. Sólo cambiaremos nuestra cartera cada cierto tiempo para minimizar costes de transacción. Justificaremos su valor en el apartado 3.2.
- Costes de transacción ("kappa"): es el coste de transacción que aplicaremos a nuestras operaciones de compra venta, sólo lo usaremos cuando apliquemos rebalanceo de cartera. Lo fijaremos en 40 puntos básicos (bps)[21].
- Ventana de estimación ("window"): es el número de datos/días del pasado que usaremos para estimar cual es la cartera óptima de inversión. Justificaremos su valor en el apartado 3.2.
- Tamaño total de la matriz histórica ("training"): es el volumen de datos total que usaremos para simular en las matrices startOneEU.m y startOneSP.m. Justificaremos su valor en el apartado 3.2.
- Valor de corte (cutoff y cutoffsmall): no ofreceremos en la cartera valores de inversión cuyo porcentaje del total sea menor que el valor de corte definido. Este valor está directamente relacionado con el tamaño de la cartera, es por ello por lo que necesitamos uno distinto para cada cartera (pequeña o grande). El valor de estos parámetros ha sido fijado de forma empírica para cada mercado y cartera.

3. Obtención de tickers.

Cada vez que nos comuniquemos con Yahoo será necesario identificar a cada empresa del índice de una manera unívoca para cada mercado, esto se consigue mediante unas abreviaciones públicas llamadas stock tickers [22].

Las empresas que componen el índice Eurostoxx 50 o el S&P 500 varían cada cierto tiempo, es por ello que es necesario obtener o actualizar de alguna manera los nombres de los tickers que las componen. Dado el número de empresas que componen el S&P 500 será especialmente importante en este mercado.

El formato del fichero de tickers será un archivo de texto (.txt) con los 50 o 500 tickers de los índices correspondientes escritos cada uno en una línea, ejemplo Tabla 3, la primera fila **Tickers** no estaría incluida en el fichero.

Tickers
ABI.BR
AI.PA
ALV.DE
ASML.AS
BAS.DE
BAYN.DE
BBVA.MC

Tabla 3. Fichero de Tickers, formato txt.

Eurostoxx

En este caso los constituyentes del Eurostoxx 50 varían cada año [23]. Al ser sólo 50 valores y variar con tanta frecuencia no se encontraron servicios web gratuitos que nos permitiesen automatizar la descarga de los tickers con Matlab, por ello será necesaria la actualización manual de este fichero (tickersEUNew.txt) con las 50 empresas cada año.

S&P 500

En este caso la empresa encargada de regular este índice es Standard & Poor's. Este tipo de empresas están constantemente eliminando y añadiendo nuevas empresas que representen bien el mercado que quieren definir. Además al ser alrededor de 500 tickers la actualización manual de este fichero es inviable.

Los ficheros por tanto serán descargados de la dirección http://finviz.com. Finviz es una web que tiene como objetivo tanto traders e instituciones de finanzas, como inversores individuales. Ofrece varias herramientas para investigación análisis y visualización de datos de finanzas. En nuestro caso realizaremos la siguiente petición vía Matlab Figura 4.

```
fileName =
urlwrite('http://finviz.com/export.ashx?v=152&f=idx_sp500&ft=1&ta=1
&p=d&r=1&c=1', ['tickersSP500.' exportFormat]);
```

Figura 4. Petición de Tickers.

El fichero devuelto, no coincidirá exactamente con nuestro formato ya que incluye una cabecera "Ticker" y añade comillas a cada ticker por lo que será necesario hacerle un tratamiento para ajustarlo al nuestro, ver Tabla 4.

"Ticker"
"A"
"AA"
"AAPL"
"ABBV"
"ABC"
"ABT"
"ACE"

Tabla 4. Fichero de Tickers descargado.

4. Descarga de datos.

Una vez tenemos los tickers y hemos definido los parámetros, podemos proceder a la descarga de datos desde Yahoo. La matriz que obtengamos será nuestra matriz de mercado inicial, sobre la que haremos la mayoría de las operaciones. La función que usamos para ello estará en matrixInit.m Figura 5.

```
[tickersSP,stocksSP,datesSP]=matrixInit(training,fileSP500);
```

Figura 5. Rutina de obtención de la matriz inicial.

Esta función realizará no sólo la descarga sino también una limpieza inicial de las matrices devueltas por Yahoo eliminando las empresas que no sean adecuadas para nuestras rutinas de decisión. Ver apartado siguiente.

Los parámetros de entrada serán el nombre del archivo de todos los tickers (fileSP500), y el tamaño de la matriz que queremos tanto para el Eurostoxx como para el S&P 500 el tamaño de "training" es de 3000 días.

Las salidas de esta función son las siguientes:

- *stocksSP*: **matriz de trabajo o de mercado** con m filas y n columnas. Las columnas representan el número de días y las filas las empresas. Para ambos mercados tendremos n=3000 días de datos ya limpios, sin embargo el número de empresas variará para cada mercado siendo, en el caso de Eurostoxx en torno a m=45-50 empresas y para el S&P entre m=450-500 empresas.
- *tickersSP*: vector de una sola columna (tamaño m) que contendrá los nombres de los tickers constituyentes de la matriz stocksSP.
- *datesSP*: un array (tamaño m) de enteros con todas las fechas de la matriz resultante. Es importante llevar en todo momento el seguimiento de las fechas pues los días representados en la matriz de trabajo los días no van a ser consecutivos pero sí ordenados.

En petición que haremos a Yahoo incluiremos la fecha de inicio, la fecha final (hoy), la empresa de la cual queremos los datos y la frecuencia de los mismos (diarios).
Para calcular la fecha de inicio realizamos el siguiente cálculo. Necesitamos una matriz de mercado de 3000 valores, 1700 datos de entrenamiento más 5 años de datos(~1300) hasta hoy. Tomando en cuenta que un año son 252 días cotizados y que Yahoo puede contener errores en determinados días estimamos que cada año tendrá al menos 240 datos, por tanto si queremos 3000 datos (numColumnas) la fecha de inicio (startDate) se calculará de la siguiente manera a partir de hoy (endDate), Figura 6.

```
startDate=endDate-ceil(numColumnas/240)*365;
```

Figura 6. Fecha inicial, rutinas históricas.

La petición que haremos para obtener el archivo .csv será una petición http GET a Yahoo con el siguiente formato Figura 7.

```
% download historical data using the Yahoo! Finance website
[temp, status] = urlread(strcat('http://ichart.finance.yahoo.com/table.csv?s='...
    ,tickers{i},'&a=',bm,'&b=',bd,'&c=',by,'&d=',em,'&e=',ed,'&f=',...
    ey,'&g=',freq,'&ignore=.csv'));
```

Figura 7. Petición GET, rutinas históricas.

Tendremos que iterar esta operación por cada uno de los tickers y el archivo que recibiremos tendrá formato .csv con los datos ordenados por filas y columnas y con una coma como delimitador ver Figura 8.

```
1  Date,Open,High,Low,Close,Volume,Adj Close
2  2014-01-27,57.90,58.37,56.98,58.30,4669000,58.30
3  2014-01-24,59.22,59.46,57.71,57.87,2656300,57.87
4  2014-01-23,60.48,60.51,59.41,59.47,3063500,59.47
5  2014-01-22,60.82,61.22,60.71,60.93,2769400,60.93
6  2014-01-21,60.69,60.88,59.86,60.85,3008500,60.85
7  2014-01-17,60.83,60.83,60.33,60.71,1699500,60.71
8  2014-01-16,60.24,60.50,60.04,60.50,1836000,60.50
```

Figura 8. Archivo recibido de Yahoo Finance.

5. Limpieza y ajuste de matrices.

Uno de las principales dificultades de este proyecto radica en que los datos que manejaremos son públicos y la empresa proveedora es externa al proyecto (Yahoo). Por ello no tendremos nunca control total sobre los datos que obtendremos.

Como se puede observar en la Figura 8 las fechas suministradas no son consecutivas y no tendremos control sobre posibles errores o incoherencias al recibir estos ficheros por parte de Yahoo. A lo largo de todo el proyecto se han incluido repetidas veces lógica adicional en esta parte del código para evitar singularidades y salvar el mayor número de errores posibles.

Además al recibir los datos de cada empresa por separado, algunas empresas en determinadas fechas no tendrán datos que otras sí. Por ello es necesario un seguimiento meticuloso de la matriz de mercado, las fechas de Yahoo y nuestro propio vector de fechas. En el caso concreto del mercado S&P 500 este seguimiento y ordenación para cada una de las empresas resultaba particularmente pesado (~2 horas de ejecución) por lo que también se decidió invertir tiempo en diversas optimizaciones del algoritmo de recepción y reordenado hasta conseguir un tiempo de ejecución razonable (~15 minutos).

La matriz de mercado tras el reordenamiento queda de la siguiente manera, ver Figura 9.

15	73.3200	68.4300	67.5300	0	0	67.3100	69.5900
16	1.4700	1.4700	1.5500	0	0	1.5500	1.5400
17	15.2200	15.5700	15.8300	0	0	15.7900	15.8000
18	24.6400	24.1400	23.4100	0	0	23.1400	23.6100
19	0	0	0	0	0	0	0
20	0.5900	0.5800	0.5700	0	0	0.5700	0.5800
21	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0
23	2.2200	2.2000	2.1800	0	0	2.2200	2.2100
24	3.4200	3.3300	3.3700	0	0	3.4800	3.4800
25	25.9700	25.7400	25.5400	0	0	25.6400	25.7400
26	49.8600	48.0300	47.6200	0	0	48.1700	47.6200
27	105.0400	105.9400	105.4600	0	0	106.4900	108.4800
28	0.1600	0.1700	0.1600	0	0	0.1600	0.1800
29	0	0	0	0	0	0	0

Figura 9. Matriz de mercado tras reordenar.

Como se puede ver esta matriz contiene numerosos valores nulos ya sea por errores en los datos de Yahoo o porque es fin de semana o días festivos. El primer tratamiento que haremos será eliminar las columnas (o días) en las que todos los valores son cero asumiendo que estos son días de no cotización, eliminando estos días también del vector de fechas, ver Figura 10.

15	73.3200	68.4300	67.5300	67.3100	69.5900	69.0700	69.5200
16	1.4700	1.4700	1.5500	1.5500	1.5400	1.5300	1.5500
17	15.2200	15.5700	15.8300	15.7900	15.8000	15.6400	15.3300
18	24.6400	24.1400	23.4100	23.1400	23.6100	22.6500	22.8200
19	0	0	0	0	0	0	0
20	0.5900	0.5800	0.5700	0.5700	0.5800	0.5700	0.5800
21	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0
23	2.2200	2.2000	2.1800	2.2200	2.2100	2.1600	2.1200
24	3.4200	3.3300	3.3700	3.4800	3.4800	3.4000	3.4200
25	25.9700	25.7400	25.5400	25.6400	25.7400	25.6700	25.6400
26	49.8600	48.0300	47.6200	48.1700	47.6200	47.4800	47.5800
27	105.0400	105.9400	105.4600	106.4900	108.4800	106.8500	107.4500
28	0.1600	0.1700	0.1600	0.1600	0.1800	0.1800	0.1900
29	0	0	0	0	0	0	0

Figura 10. Matriz de mercado sin días no cotizados.

Por último eliminaremos aquellos valores que pueden introducir comportamientos singulares a la hora de aplicar nuestras estrategias. En la Figura 10 se puede ver como hay algunas empresas de las que Yahoo no dispone datos durante un largo periodo de tiempo, ya sea porque son nuevas y no cotizaban hace 5 años o por pérdida de datos. Eliminaremos estas empresas de nuestra matriz de mercado al no poder predecir con exactitud su comportamiento posteriormente en nuestras estrategias. Si la empresa tiene un vacío de 1 mes de datos, será eliminada de nuestra matriz de predicción o mercado. La matriz limpia finalmente tendrá el siguiente aspecto Figura 11.

15	73.3200	68.4300	67.5300	67.3100	69.5900	69.0700	69.5200
16	1.4700	1.4700	1.5500	1.5500	1.5400	1.5300	1.5500
17	15.2200	15.5700	15.8300	15.7900	15.8000	15.6400	15.3300
18	24.6400	24.1400	23.4100	23.1400	23.6100	22.6500	22.8200
19	0.5900	0.5800	0.5700	0.5700	0.5800	0.5700	0.5800
20	2.2200	2.2000	2.1800	2.2200	2.2100	2.1600	2.1200
21	3.4200	3.3300	3.3700	3.4800	3.4800	3.4000	3.4200
22	25.9700	25.7400	25.5400	25.6400	25.7400	25.6700	25.6400
23	49.8600	48.0300	47.6200	48.1700	47.6200	47.4800	47.5800
24	105.0400	105.9400	105.4600	106.4900	108.4800	106.8500	107.4500
25	0.1600	0.1700	0.1600	0.1600	0.1800	0.1800	0.1900
26	1.3389e+04	1.3211e+04	1.3058e+04	1.2777e+04	1.2777e+04	1.2649e+04	1.2803e+04
27	55.4100	55.1700	53.8300	53.8300	53.7500	53.7900	53.3600
28	283.8600	283.2300	283.0300	287.2700	290.5700	284.7900	287.1000
29	68.2200	65.7600	64.9800	64.9400	66.7400	67.2400	67.3200

Figura 11. Matriz de precios histórica limpia.

Esta matriz aún puede contener valores nulos en zonas puntuales y será necesario rellenar y estimar de alguna manera estos "*missing values*". Si bien, este nuevo ajuste se realizará más adelante en las rutinas de estrategias y se explica en el apartado 7. Aplicación de estrategias y simulación histórica.

6. Descarga y cambio de moneda.

Ya disponemos de una matriz de mercado con la que podemos trabajar. La moneda de esta matriz serán dólares o euros dependiendo del mercado S&P 500 o Eurostoxx. Procederemos ahora al cambio de divisa de la matriz de mercado.

```
[exchange outputWeb]=getEuroDolar(datesSP);
```

Figura 12. Obtención de datos del cambio de moneda.

Para simplificar el código usaremos la función getEuroDolar(dates), que recibirá cómo parámetro el vector de fechas de la matriz de mercado("*datesSP*"), y nos devolverá otro vector ya ordenado y de la misma longitud que el vector de fechas de la matriz de mercado preparado ya para operar con él ("*exchange*"). La otra variable de salida ("*outputWeb*") son los datos tal y cómo son enviados por la página web de dónde son obtenidos.

La descarga de datos se realizará también de forma gratuita de la página de la reserva federal del banco de ST. Louis [24] Figura 13.

```
url='https://research.stlouisfed.org/fred2/series/DEXUSEU/downloaddata';
fileName = urlwrite(url,['test.' exportFormat],'post',...
    {'_qf__mainform','','native_frequency','Daily',...
    'units','lin','frequency','Daily','aggregation',...
    'Average','obs_start_date',startDate','obs_end_date',...
    endDate','file_format','csv','download_data','Download+Data'});
```

Figura 13. Descarga de cambio de moneda.

Será necesario especificar varios parámetros:

- Rango de fechas: primera y última posición de nuestro vector de fechas, recibido por parámetro en la función.
- Frecuencia de los datos: diario.
- Formato de texto: separado por coma (.csv).
- Tipo de cambio: dólares americanos a un euro.

El fichero recibido será cómo el de la figura 14.

```
1  DATE,VALUE
2  1999-01-04,1.1812
3  1999-01-05,1.1760
4  1999-01-06,1.1636
5  1999-01-07,1.1672
6  1999-01-08,1.1554
```

Figura 14. Fichero de cambio de moneda.

Sólo ofreceremos 5 años de datos en la plataforma web, y teniendo en cuenta también el tamaño de entrenamiento como mucho necesitaremos datos de hace 7 años lo que nos evita problemas de cambios de moneda en Europa al trabajar todas las empresas de los mercados elegidos en euros o dólares.
El fichero recibido será tratado en la función getEuroDolar(dates) para ajustar los datos de cambio a los de la matriz de mercado.

Dependiendo del mercado multiplicaremos o dividiremos cada fila de la matriz de mercado por el vector de cambio, obteniendo una nueva matriz del mercado con la divisa cambiada.

7. Aplicación de estrategias y simulación histórica.

A partir de este punto disponemos de dos matrices del mismo mercado listas para ser utilizadas en nuestras estrategias. Como ya se ha comentado tanto para S&P 500 como para el Eurostoxx 50 tendremos 4 carteras para cada uno (8 en total) ver Figura 15. Y será necesario simular históricamente hacia atrás cada una de estas 4 carteras.

```
[ stratReturnsSP w1 daysUntilRebalance]=fstratOne( stocksSP, window, rebalance,kappa, cutoff,kmaxStocks);
[ stratReturnsSPeuro w2 lD2]=fstratOne( SPeuro, window, rebalance,kappa, cutoff,kmaxStocks);
[ stratReturnsSPsmall w3 lD3]=fstratOne( stocksSP, window, rebalance,kappa, cutoffSmall,kmaxStocksSmall);
[ stratReturnsSPeurosmall w4 lD4]=fstratOne( SPeuro, window, rebalance,kappa, cutoffSmall,kmaxStocksSmall);
```

Figura 15. Rutina de obtención de retornos. Cuatro carteras.

La función fstratOne(...) recibirá los siguientes parámetros ver Figura 15.

- *stocks*: matriz de mercado, ya sea en euros o dólares.
- *window*: ventana de entrenamiento, en días.
- *rebalance*: periodo entre rebalanceo de carteras, en días.
- *kappa*: costes de transacción, en %
- *cutoff*: valor umbral que nos fijará el tamaño de la cartera, en %. Por debajo de este % no elegiremos stocks.
- *kmaxStocks*: tamaño máximo de empresas en la cartera.

Los datos de salida una vez ejecutemos estas funciones serán:

- *stratReturns*: vector de retornos históricos para el mercado, cartera y divisa elegida.
- *weights*: matriz de pesos históricos de tamaño mxn igual al de la matriz de mercado introducida.
- *daysUntilRebalance*: días hasta el próximo rebalanceo de cartera. Este dato es necesario más adelante para las funciones de actualización.

El flujo de trabajo que realizaremos en esta función será el siguiente Figura 16.

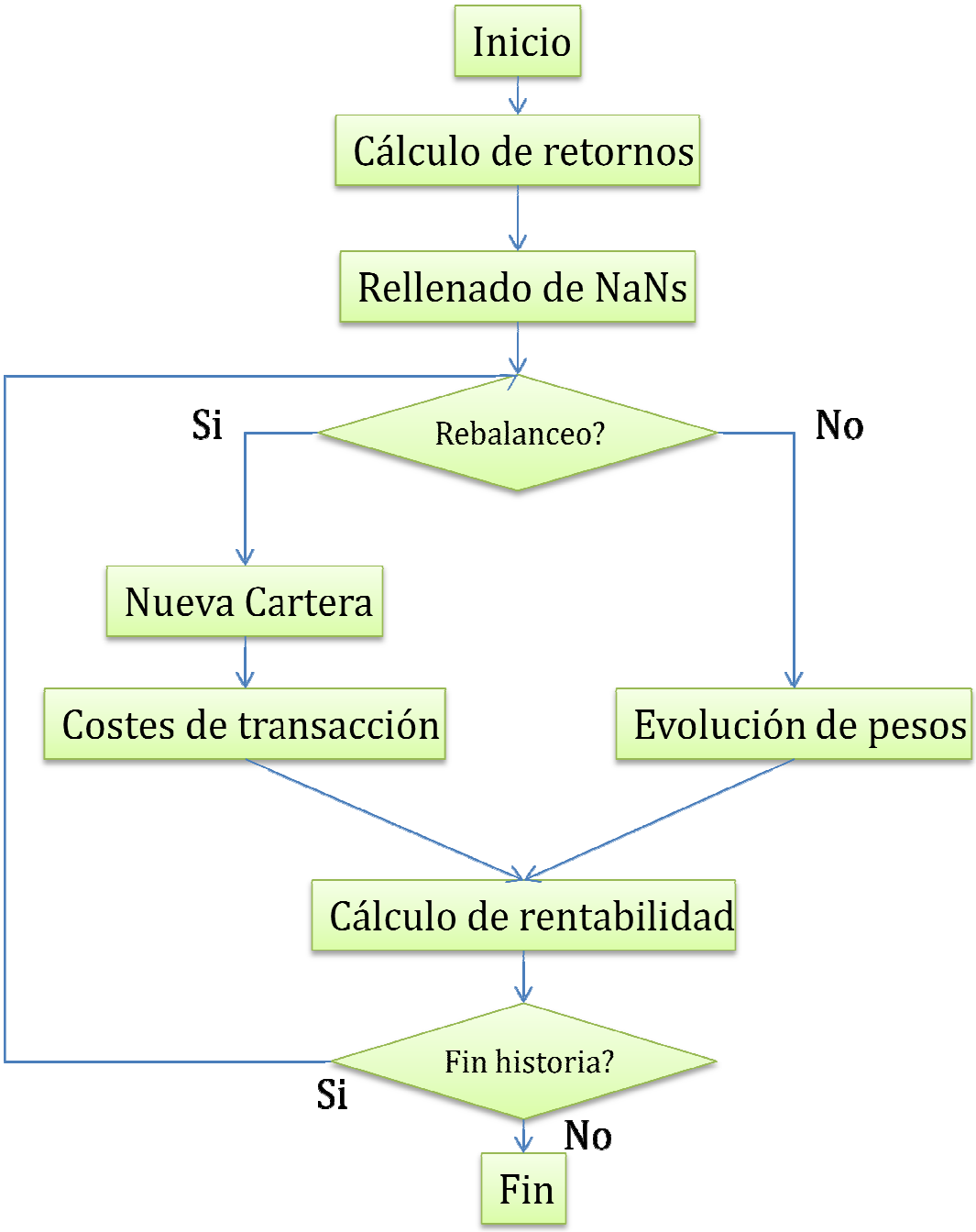

Figura 16. Flujo de trabajo función fstratOne.m

El primer paso que realizamos es la conversión de los precios en la matriz de mercado que son 0 a NaN, para que cuando calculemos los retornos no tome los valores 0 como precio, ver Figura 17.

	1	2	3	4
1923	33.3000	52.6800	73.2600	37.7300
1924	33.5000	53.7600	75.2500	38.1900
1925	33.6000	53.7600	76.1300	39.3500
1926	33.6600	53.9000	76.3800	38.9600
1927	33.7000	53.8400	NaN	38.9200
1928	NaN	NaN	NaN	38.9200
1929	34.2000	54.2000	77.0600	38.8800
1930	34.5600	53.8000	76.6300	38.8800
1931	34.3100	53.7300	76	38.8800
1932	34.4500	53.4900	NaN	38.9600
1933	NaN	NaN	NaN	38.9600
1934	35.0900	54.4400	77.2200	39.3800
1935	34.6500	53.3500	77.4500	39.4600

Figura 17. Matriz de precios con NaN.

La matriz del mercado a tratar refleja la evolución histórica de precios, sin embargo las rutinas de estrategias trabajan con retornos y no con precios. Además a partir de ahora traspondremos la matriz de mercado y cada columna corresponderá a una empresa siendo las filas los días cotizados ver Figura 17. Para calcular los retornos usaremos la siguiente función de Matlab ver Figura 18.

```
returnsAux=price2ret(R,[],'Periodic');
```

Figura 18. Obtención de retornos a partir de precios.

Si "*p*" es la serie completa de precios de entrada, cada salida simple "*r(i)*" se puede calcular de esta manera r(i) = [p(i+1)/p(i)-1].Partiendo de la Figura 17 podemos ver como quedaría la operación en la Figura 19.

	1	2	3	4
1923	0.0060	0.0205	0.0272	0.0122
1924	0.0030	0	0.0117	0.0304
1925	0.0018	0.0026	0.0033	-0.0099
1926	0.0012	-0.0011	NaN	-0.0010
1927	NaN	NaN	NaN	0
1928	NaN	NaN	NaN	-0.0010
1929	0.0105	-0.0074	-0.0056	0
1930	-0.0072	-0.0013	-0.0082	0
1931	0.0041	-0.0045	NaN	0.0021
1932	NaN	NaN	NaN	0
1933	NaN	NaN	NaN	0.0108
1934	-0.0125	-0.0200	0.0030	0.0020
1935	-0.0072	-0.0062	0.0077	0.0106

Figura 19. Matriz de retornos con NaN.

Rellenado de NaNs.

Antes de aplicar las rutinas de estrategias aplicaremos la función "*fillNaN(returns)*" para rellenar estos NaN o "*missing values*" en nuestra Figura 19. Esta función fue proporcionada por los tutores del proyecto, por lo que sólo se dará una breve explicación sobre su funcionamiento sin entrar en un extenso detalle.

La entrada de esta matriz será una matriz de retornos de dimensión TxN, donde T es el número de observaciones o días cotizados y N es el número de activos o stocks. La salida será dicha matriz sin NaNs, ver Figura 20.

	1	2	3	4
1923	0.0060	0.0205	0.0272	0.0122
1924	0.0030	0	0.0117	0.0304
1925	0.0018	0.0026	0.0033	-0.0099
1926	0.0012	-0.0011	-9.3360e-04	-0.0010
1927	-2.3351e-04	-1.2252e-04	-0.0015	0
1928	0.0018	0.0032	0.0035	-0.0010
1929	0.0105	-0.0074	-0.0056	0
1930	-0.0072	-0.0013	-0.0082	0
1931	0.0041	-0.0045	-4.1970e-05	0.0021
1932	-2.3351e-04	-1.2252e-04	-0.0015	0
1933	0.0050	0.0082	0.0112	0.0108
1934	-0.0125	-0.0200	0.0030	0.0020
1935	-0.0072	-0.0062	0.0077	0.0106

Figura 20. Matriz sin NaNs.

La rutina calculará el valor esperado dado por un modelo factorial de mercado, de manera que para el cálculo de NaNs no sólo tendremos en cuenta el valor individual de cada activo y su historia sino que dicho valor tendrá en cuenta además el comportamiento general del mercado esos días. Básicamente buscaremos todas aquellas rentabilidades $R_{i,t}$=NaN y le aplicaremos el siguiente modelo factorial:

$$R_{i,t} = \alpha + \beta R_{mkt} + \varepsilon_t$$

Estimamos el modelo factorial descrito mediante Mínimos Cuadrados Ordinarios (MCO). Por último, utilizamos el modelo estimado para rellenar aquellos valores faltantes en nuestro conjunto de datos.

Cálculo de historia.

En este punto la matriz está lista para entrar en las rutinas de estrategias. Cómo necesitamos un primer volumen de datos para ser mostrado en la web necesitaremos iterar históricamente aplicando nuestras estrategias en los últimos 5 años.

El primer paso será obtener la matriz de entrenamiento "*training*" en función de la ventana elegida y la iteración en la que estemos. Esta matriz será la que usaremos como entrada en nuestra predicción de cartera.

Las dos siguientes funciones que atienden al cálculo de estrategias "*ShrinkMidentity(...)*" y "*LowVolPortfolio(...)*" ver figura 21, han sido suministradas por los Tutores del proyecto. Para una explicación más detallada de su funcionamiento se puede acudir a la siguiente bibliografía [25][26][27][28]. Además en el apartado 3.2 Se ofrece una descripción general sobre su funcionamiento.

```
[SigmaMidentity]=ShrinkMidentity(training);
[w1, exitflag1] = LowVolPortfolio(SigmaMidentity,cutoff,kmax);
```

Figura 21. Rutinas de estrategias.

La justificación del uso de esta estrategia frente a las 3 disponibles se realizará en el apartado 3.2.La entrada que tendrá será tanto la matriz de retornos "*training*" como las variables recibidas por parámetro:

- cutoff: valor umbral que nos fijará el tamaño de la cartera, en %. Por debajo de este % no elegiremos stocks.
- kmax: tamaño máximo de empresas en la cartera.

La salida que obtendremos será un vector de pesos "*w1*" de tamaño N igual al número de activos (columnas) de la matriz de retornos, donde podremos ver el ratio de inversión para cada activo ver Figura 22.

	1
1	0.0960
2	0.0429
3	0
4	0
5	0
6	0
7	0
8	0
9	0.0426
10	0
11	0.0600
12	0

Figura 22. Vector de pesos.

Cada peso representa el porcentaje invertido de nuestro presupuesto en cada uno de los activos, por ello la suma de los elementos de este vector será siempre 1.

De forma general, a partir de este vector y multiplicando por los retornos del día siguiente obtendremos un vector de rentabilidades para cada día "*rentc1(i,:)*".
Sin embargo hay que particularizar según sea el caso, dentro del bucle podemos diferenciar entre 3 estados diferentes:

- Primera Iteración.
- Día de no rebalanceo.
- Día de rebalanceo.

Primera iteración

La primera vuelta del bucle será distinta al resto. Aplicamos estrategias a la primera matriz de entrenamiento, que abarcará desde la posición 1 hasta la posición coincidente con el tamaño deseado "*window*". Y obtenemos un vector o cartera de pesos "*w1*".
Hay que tener en cuenta que en esta primera iteración o compra generará unos costes de transacción igual al coste fijado de 40 puntos básicos (kappa). Para reflejar esto hacemos que la primera rentabilidad sea igual al coste de transacción.

Día de no rebalanceo

Los días de no rebalanceo no cambiaremos nuestra cartera es por ello por lo que no será necesario calcular costes de transacción pero será necesario reflejar los cambios en el mercado en nuestro portfolio. Sí ayer teníamos un 7% de nuestro

capital en Telefónica hoy ese porcentaje habrá variado con respecto al total de la cartera. Por tanto en caso de no rebalanceo realizaremos dos cosas ver Figura 23:

- Cálculo de rentabilidad con los pesos de ayer y los retornos de hoy.
- Evolución de nuestro portfolio con los pesos de ayer y retornos de hoy.

```
%Evolución de pesos
wbh1= weights1(i-1,:).*(1+returns(window+i-1,:));
%Reescalado de pesos
wbh1=wbh1./(sum(wbh1));
%Almacenamiento en la matriz histórica de pesos
weights1(i,:) = wbh1;
%Rentabilidad parcial para cada valor
rent1(i,:)=weights1(i-1,:).*returns(window+i-1,:);
%Rentabilidad total
rentc1(i,:)=sum(rent1(i,:));
```

Figura 23. Evolución de pesos y cálculo de rentabilidades. Día de no rebalanceo.

Día de rebalanceo

Los días de rebalanceo cambiaremos nuestra cartera de recomendación de forma parcial o total. Lo primero que haremos será aplicar las rutinas de estrategia a la matriz de entrenamiento que corresponda al día que estamos simulando. Una vez calculados los nuevos pesos será necesario realizar las siguientes operaciones, ver Figura 24:

- Cálculo de rentabilidad con los pesos de ayer y los retornos de hoy. Hasta el próximo día de simulación no usaremos los nuevos pesos.
- Evolución de nuestro portfolio con los pesos de ayer y retornos de hoy. Esta parte solo es necesaria para calcular los costes de transacción
- Cálculo de la norma 1. Es una medida de lo que cambian nuestros nuevos pesos respecto a los antiguos
- Descuento de los costes de transacción sólo para la parte de la cartera que es necesario cambiar.

```
%Rentabilidad parcial para cada valor
rent1(i,:)=weights1(i-1,:).*returns(window+i-1,:);
%Evolución de pesos
wbh1= weights1(i-1,:).*(1+returns(window+i-1,:));
%Reescalado de pesos
wbh1=wbh1./(sum(wbh1));
%Cálculo de la norma 1
norm11=abs(weights1(i,:)-wbh1);
%Descuento de costes de transacción
rentc1(i,:)=(1+sum(rent1(i,:)))*(1-kappa*sum(norm11))-1;
```

Figura 24. Evolución de pesos y cálculo de rentabilidades. Día de rebalanceo.

Sólo tomaremos en cuenta el cambio diferencial para los costes, de manera que si tenemos un 7 % en Telefónica y la nueva recomendación ha asignado un peso de un 9% sólo aplicaremos los costes de vender ese 2% sobrante manteniendo el otro 7%.

Podemos ver un ejemplo de la evolución de pesos y carteras en la siguiente tabla, ver Figura 25.

	37	38	39	40	41	42
32	0.0793	0	0	0	0.0820	0.0837
33	0.0787	0	0	0	0.0831	0.0854
34	0.0800	0	0	0	0.0819	0.0848
35	0.0785	0	0	0	0.0822	0.0851
36	0.0774	0	0	0	0.0803	0.0845
37	0.0753	0	0	0	0.0802	0.0845
38	0.0722	0	0	0	0.0828	0.0864
39	0.0752	0	0	0	0.0805	0.0848
40	0.0508	0	0.0468	0	0	0
41	0.0514	0	0.0477	0	0	0
42	0.0514	0	0.0480	0	0	0
43	0.0514	0	0.0480	0	0	0
44	0.0514	0	0.0480	0	0	0
45	0.0517	0	0.0483	0	0	0
46	0.0513	0	0.0483	0	0	0
47	0.0514	0	0.0477	0	0	0

Figura 25. Matriz de histórico de pesos.

Las filas representan los días de cotización y las columnas los activos. La fila 1 será el primer día de cotización o el día más antiguo y la última fila coincidirá con hoy o último día cotizado.

El vector total de rentabilidad representará los retornos obtenidos cada día para la cartera total ver Figura 26. Los días representados coinciden con la matriz de pesos.

	1
1291	0.0056
1292	-0.0102
1293	-0.0247
1294	-0.0058
1295	-0.0037
1296	-0.0037
1297	0.0023
1298	-0.0045
1299	-0.0096
1300	-0.0071
1301	

Figura 26. Vector de retornos o rentabilidad.

Como se puede observar el tamaño total de nuestra rentabilidad histórica son de 1300 días cotizados. Como ya se explicó, de la matriz inicial de 3000 datos, estamos usando 1700 para entrenar. La rentabilidad mas reciente se encuentra en la fila final.

8. Descarga de datos del índice de inversión.

Para poder comparar el rendimiento de nuestras estrategias en la plataforma web ofreceremos medidas y gráficos de los índices S&P 500 y Eurostoxx. De nuevo descargaremos desde Yahoo los datos de dichos índices con los tickers ^GSPC y ^STOXX50E. Para que no haya problemas con las fechas tendremos una rutina independiente que solventará posibles problemas con los datos de Yahoo y reordenará los datos para que la matriz devuelta coincida con las fechas de la matriz de mercado ver Figura 27.

```
%Download of comparing Indices GSPC
[tickersI,stocksI,datesI]=getIndices(xdates,'gspc.txt');
```

Figura 27. Rutina de descarga de índices.

9.Descarga y cambio de moneda del índice.

Una vez descargados y reordenados de forma análoga a como operábamos en la matriz de mercado descargaremos el cambio de moneda y multiplicaremos o dividiremos el vector según corresponda, por último procederemos al cálculo de retornos con la misma función de Matlab que usamos para la matriz de mercado "*price2ret(...)*". Al sólo tener un índice no será una matriz sino un vector de

retornos del que podremos sacar posteriormente las medidas de rendimiento del índice, ver Figura 28.

	1
7	-0.0488
8	-0.0100
9	-0.0071
10	-0.0062
11	0.0284
12	-0.0222
13	-0.0473
14	-0.0096
15	0.0422
16	-0.0469

Figura 28. Vector de retornos del índice.

10.Cálculo de medidas de rendimiento.

Por último queda calcular y representar todas las medidas que serán mostradas en la web. Usaremos dos rutinas para esta última parte ver Figura 29.

```
startHistoTickers;
chartsAndData;
```

Figura 29. Rutinas de cálculo de medidas de rendimiento.

Estos scripts generarán los ficheros de la web a partir de las rentabilidades históricas obtenidas, además la rutina semanal usará también estas funciones. Debido a la interacción de estos datos con la página web, las rutinas de simulación histórica y las rutinas semanales se detallará en su propio apartado el 3.2.

3.1.2 Actualización semanal.

Con la simulación histórica realizada ya tendríamos todos los elementos necesarios para poder presentar los datos en la página web. Ahora debemos tener unas rutinas que actualicen los datos de nuestro sistema cada cierto periodo de tiempo. Dada la frecuencia con la que Yahoo actualiza sus datos no es conveniente una actualización diaria, normalmente los datos del último día de cotización son ofrecidos por Yahoo entre 1 y 3 días más tarde. Por ello actualizaremos los datos semanalmente los Domingos. Esta actualización es independiente de si es necesario un rebalanceo de nuestras carteras o no. En caso de no rebalanceo también se

actualizarán todos los datos de rendimiento así como gráficas y evolución de los pesos.

Las dos rutinas que usaremos serán *weeklyEU.m* y *weeklySP.m*, cada una para el mercado correspondiente. Para evitar perder los datos anteriores tendremos dos carpetas separadas para cada mercado donde realizaremos un pequeño backup con los datos almacenados hasta ahora, los datos antiguos serán almacenados en las carpetas "*webInfoOneEUOld*" y "*webInfoOneSPOld*". Este backup tendrá como objetivo asegurar la atomicidad del sistema, en caso de imprevistos en la función se restaurarán todos los datos a su estado tal y como estaba la semana anterior ver Figura 30.

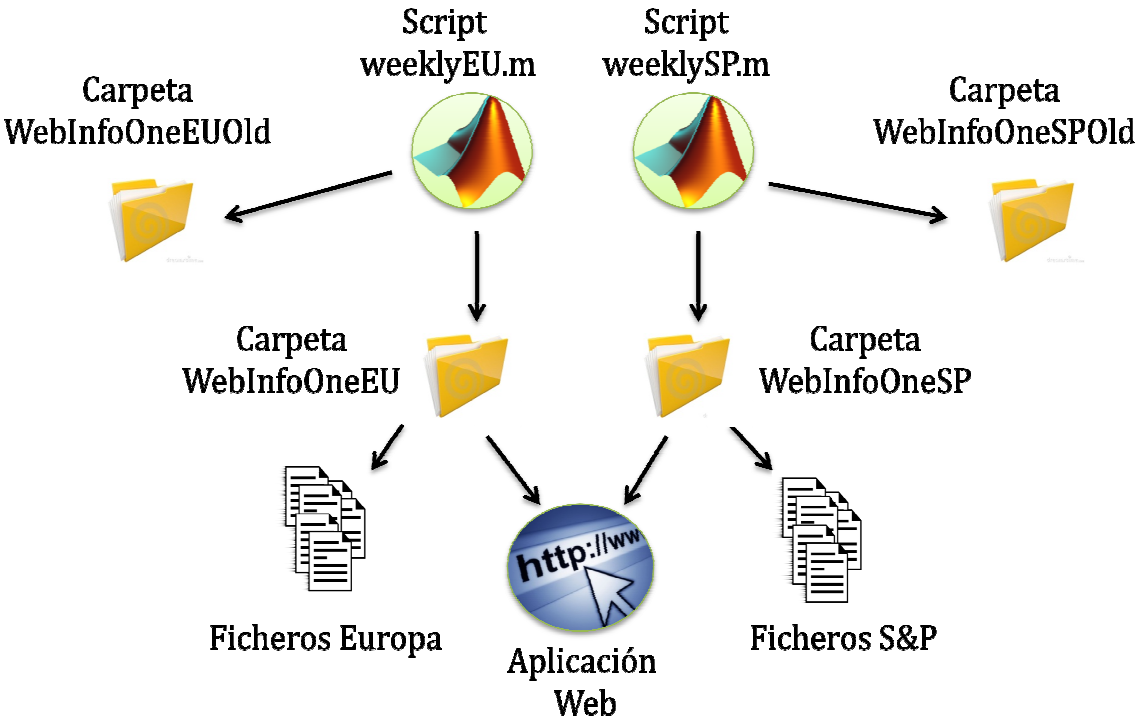

Figura 30. Esquema rutinas semanales.

Cada una de las rutinas actualizará sus 4 carteras según el mercado siguiendo en orden las siguientes operaciones.

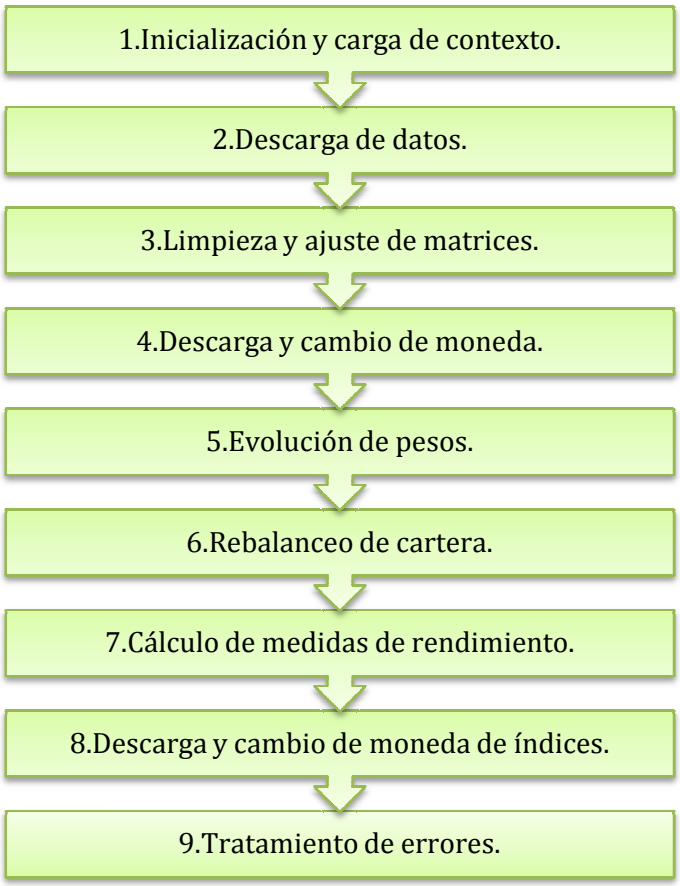

Figura 31. Operaciones rutinas semanales.

Algunas de las funciones realizadas serán muy similares a las realizadas en las rutinas de simulación histórica, por ello no serán definidas de forma tan exhaustiva en este apartado.

1.Inicialización y carga de contexto.

Lo primero que haremos será especificar el nombre de la carpeta donde vamos a realizar el backup y la carpeta dónde están actualmente los datos. Todo el contexto de las variables de Matlab está guardado en un archivo llamado "*weeklyInfo.mat*" tanto las rutinas de simulación histórica como las semanales una vez acabado todas las operaciones guardan en ese fichero el contexto. Tendremos dos ficheros de contexto uno para cada mercado y cada uno de ellos contendrá los datos de las 4 carteras de inversión.

2.Descarga de datos.

Una vez cargado todo el contexto podemos comenzar a la descarga de datos. A diferencia de las rutinas de simulación histórica aquí no descargaremos nuevos ficheros de tickers sino que usaremos los antiguos. Aunque sólo se esté invirtiendo en por ejemplo 6 empresas esa semana es conveniente que descarguemos todas las empresas de la matriz de mercado, debido a posibles errores en los datos de Yahoo,

podemos volver a encontrar "*missing values*" y será necesario volver a aplicar las rutinas de rellenado de NaNs que como ya se explicó tienen en cuenta el comportamiento de todo el mercado.
La fecha de inicio en este caso será calculada a partir de la última fecha en la que tenemos datos. Cómo tenemos cargado todo el contexto de la semana anterior obtener esta fecha es muy simple ver Figura 32. Además de nuevo con motivo de optimizar las rutinas de rellenado de NaNs obtendremos datos de al menos 2 meses antes a esta fecha (60 días).

```
startDate=xdates(end);
startDate=startDate-60;
```

Figura 32. Fecha inicial rutinas semanales.

La fecha final será el día en que estemos ejecutando la rutina. Esta vez a diferencia de la simulación histórica queremos preservar exactamente las mismas empresas que usamos la semana anterior por lo que llamamos directamente a la rutina de la Figura 33.

```
try
[tickers matrizyahoo] = stock_yahoo(startDate1,endDate1,currentStocks);
catch e
[tickers matrizyahoo] = stock_yahoo(startDate1,endDate1,currentStocks);
end
```

Figura 33. Rutina semanal de obtención de datos.

El try catch utilizado es necesario para solventar un error en la versión de *Matlab 2010a* con la función "*datenum()*" ver bibliografía para mas detalles [29]. La salida de la función será tanto la matriz de mercado cómo los tickers obtenidos del fichero de texto en formato Matlab.
La variable "*currentStocks*" es una cadena de texto con el nombre y ubicación del fichero con las empresas que teníamos en la matriz de mercado de la semana anterior, con el formato de la Figura 34.

```
1  A
2  AA
3  AAPL
4  ABC
5  ABT
6  ACE
7  ACN
8  ACT
```

Figura 34. Fichero con los Tickers de la última semana.

La descarga de stocks se realizará de la misma url y de forma análoga a cómo se realizaba en la rutina de simulación histórica.

3.Limpieza y ajuste de matrices.

Comprobaremos que en la nueva matriz no falte ninguno de los valores antiguos y procederemos a su limpieza. De manera análoga a la explicada en el apartado 3.1.1.5 de este documento limpiaremos las columnas de los días que no contengan datos. Sin embargo a diferencia de antes ahora no eliminaremos ninguna de las empresas (o filas), pues las que no eran aptas ya fueron eliminadas y de esta manera preservaremos las mismas empresas de la última cartera ofrecida.

4.Descarga y cambio de moneda.

En esta parte de la rutina también se procederá de la misma manera que en el apartado 3.1.1.6 usando exactamente la misma función "*getEuroDolar.m*". Se descargan los datos para las últimas fechas, y se multiplica o divide cada fila por el cambio según si es la rutina del Eurostoxx o del S&P 500 ver Figura 35.

```
[exchange outputWeb]=getEuroDolar(newDates);
for i=1:numNewTickers
    newPricesAlt(i,:)=newPrices(i,:)./exchange;
end
```

Figura 35. Bucle de cambio de moneda.

5.Evolución de pesos.

Dado que ahora no es necesaria una iteración histórica se usará una rutina nueva respecto a las de simulación histórica "*fstratOne.m*" para calcular tanto las rentabilidades cómo la evolución de los pesos. La rutina de evolución de pesos "*evolveWeights.m*" seguirá una lógica similar a la que explicamos en el apartado 3.1.1.7. De nuevo deberemos aplicarla a nuestras 4 carteras y evolucionar sus pesos.
La función tiene las siguientes entradas y salidas:

```
[returns weights] = evolveWeights(oldReturns,oldWeights,newPrices, firstDayPosition)
```

Figura 36. Rutina semanal de evolución de pesos.

Parámetros de entrada

- *oldReturns*: son los retornos de nuestra estrategia, no los de la matriz de mercado, sólo los usaremos para concatenar las nuevas rentabilidades de la última semana.

- *oldWeights*: matriz con la lista de pesos antiguos. De esta matriz sólo necesitamos los pesos del último día para evolucionarlos.
- *newPrices*: matriz de mercado de precios con los últimos días. A partir de ella calcularemos las nuevas rentabilidades o retornos de la última semana.
- *firstDayPosition:* la matriz nueva de mercado que pasamos "*newPrices*" contiene 2 meses de datos extra, aquí indicamos dónde empieza el primer día de datos nuevos.

Parámetros de salida

- *returns*: nueva matriz de retornos o rentabilidades contiene los históricos y los nuevos concatenados al final.
- *weights*: nueva matriz de pesos evolucionados, no contiene los pesos antiguos.

El flujo de trabajo que realizaremos en esta función será el siguiente Figura 37

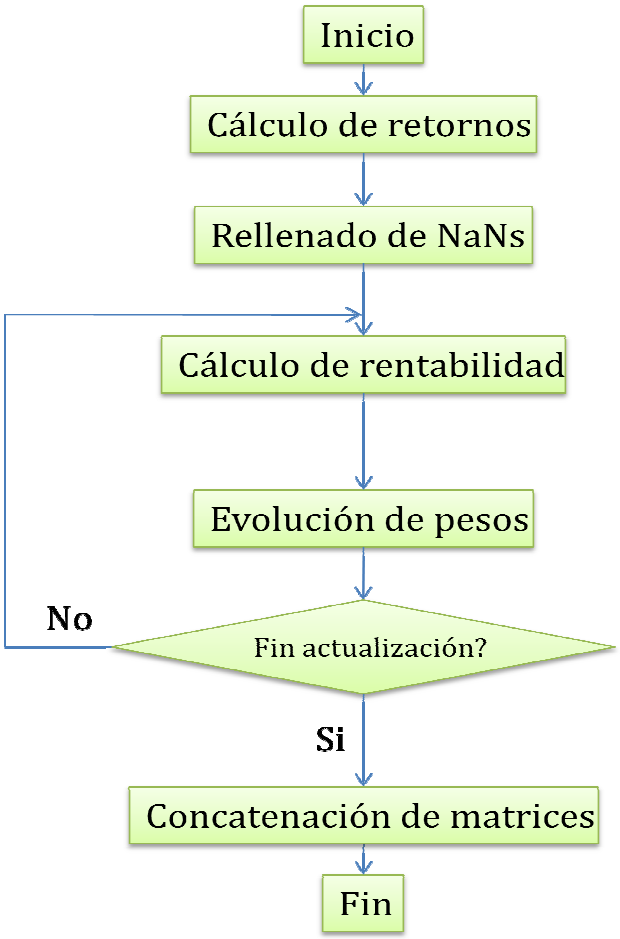

Figura 37. Flujo de trabajo función evolveWeights.m

Las matrices de mercado en ambas monedas como pasaba en la simulación histórica aún pueden contener valores nulos en lugares aislados por lo que antes de calcular los retornos cambiaremos estos nulos por NaNs tal y cómo hacíamos en las

rutinas de simulación histórica. Seguidamente aplicaremos la misma rutina de rellenado de NaNs explicada en el apartado 3.1.1.7.
Ahora cuando entramos al bucle no aplicaremos estrategias ni calcularemos nuevas carteras. Aplicaremos sólo la parte de evolución de los pesos y cálculo de rentabilidades igual que hacíamos antes.
Finalmente calcularemos los retornos totales para cada día sumando todas las rentabilidades individuales de cada activo y concatenaremos estos nuevos retornos con los históricos que pasábamos por parámetro, ver Figura 38.

```
totalReturn=sum(newRent,2);
returns=[oldReturns;totalReturn];
weights=newWeights;
```

Figura 38. Final de la rutina evolveWeights.m

6.Rebalanceo de cartera.

Hasta ahora en la actualización semanal no hemos tenido en cuenta si es necesaria o no es necesaria evaluación de cartera, esto es así para preservar la coherencia de la información ofrecida por la página web. Al hacer la actualización del servidor los Domingos un usuario que entre a lo largo de la siguiente semana siempre va a ver las carteras ofrecidas del último Domingo así que todas las rentabilidades calculadas por nuestro sistema deben coincidir con la cartera que estamos ofreciendo en ese momento.
Es por esto que hasta ahora solo hemos evolucionado los pesos y en este momento es cuando comprobamos si debemos rebalancear carteras o no. Recordemos que el periodo de rebalanceo fijado es cada 2 meses o 40 días cotizados aproximadamente y la variable que usamos para llevar el seguimiento de días que faltan es *"daysUntilRebalance*". Si este valor es nulo o negativo entonces procederemos con este paso.

En el caso de ser necesario un rebalanceo estos son los pasos que haremos ver Figura 39.

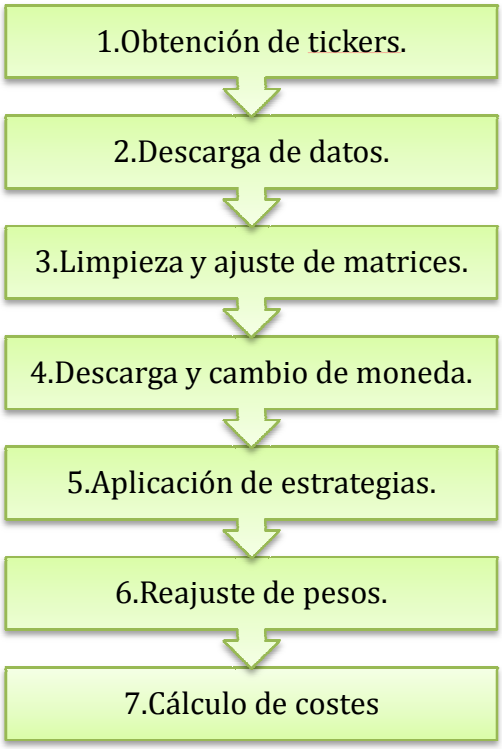

Figura 39. Operaciones semanales si existe rebalanceo.

Los 5 primeros pasos son inmediatos y análogos a la simulación histórica. La única aclaración necesaria es que el volumen de datos que descargados ahora servirá únicamente para calcular una cartera por lo que el tamaño de datos que necesitaremos será igual al del tamaño de la ventana de estimación, 1700 días hasta hoy.

Al haber descargado nuevos tickers y nuevos datos, las empresas figurantes y la posición de ellas pueden haber cambiado completamente de una semana para otra. Deberemos por tanto prestar especial atención para mantener la coherencia de los datos. Básicamente tendremos un nuevo vector de tickers, y un nuevo vector de pesos para la próxima semana. De cara a las rentabilidades sólo tendremos que reflejar los costes del cambio de cartera y será esta misma función como ya se ha visto la que se encarga de evolucionar los nuevos pesos la próxima semana.

Para saber que empresas cambian, cuales se mantienen y donde están ubicadas se utilizará una nueva rutina llamada "*reArrangeWeights.m*" con los siguientes parámetros de entrada y salida ver Figura 40.

```
[weightsToSell newWeights] = reArrangeWeights(wLast,newTickers,oldTickers )
```

Figura 40. Rutina de recolocación de tickers.

Parámetros de entrada:

- *wLast*: pesos antiguos.
- *newTickers*: tickers de las empresas de la nueva matriz de mercado que coinciden con la posición de los nuevos pesos.
- *oldTickers*: tickers antiguos, que coinciden con la posición de los últimos pesos.
-

Parámetros de salida:

- *weightsToSell*: vector de pesos antiguos.
- *newWeights*: pesos antiguos recolocados de acuerdo con la nueva estructura.

El algoritmo crea un vector de pesos vacío de tamaño igual al número de tickers de la nueva matriz. Iterando sobre cada uno de estos tickers, comprobaremos si existía ya antes. En caso de no encontrarlo no se hace nada, pero si ya teníamos esas empresa cogeremos su valor y lo colocaremos en el nuevo sitio ver Figura 41.

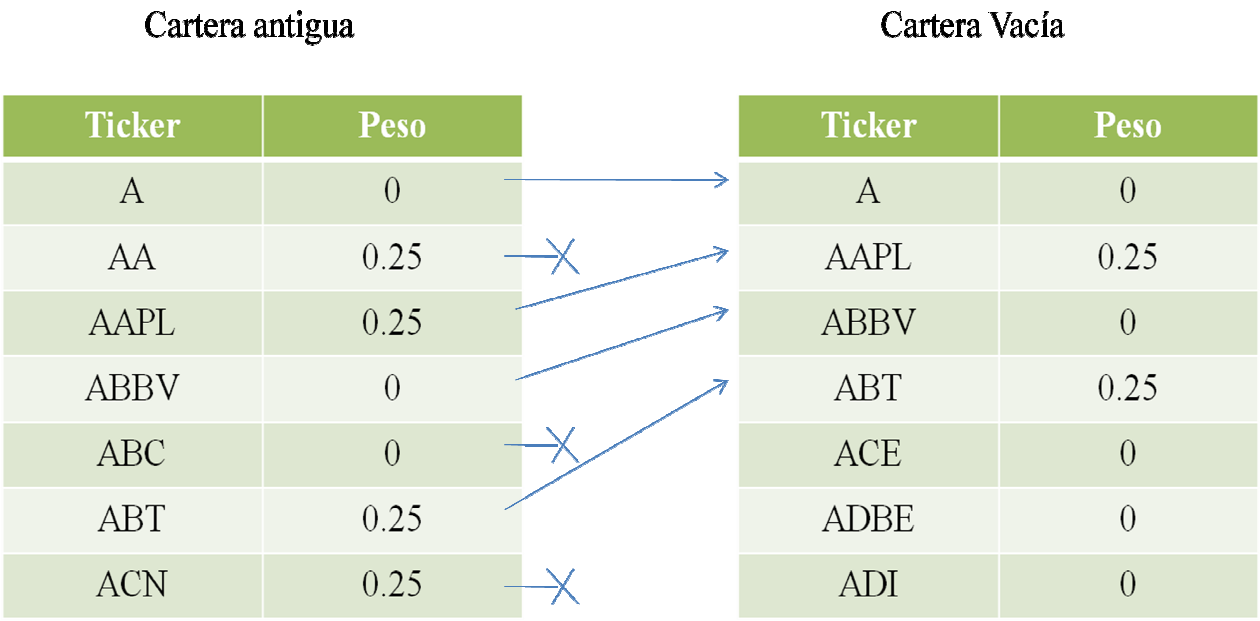

Figura 41. Esquema de reordenación de pesos y Tickers.

El caso mostrado anteriormente es un caso ficticio no son valores reales. La cartera vacía finalmente contendrá los pesos antiguos reordenados de acuerdo a la nueva matriz. Este paso nos servirá para calcular los costes de cambio de cartera. De la misma manera que procedíamos en la simulación histórica de nuevo calculamos la norma uno de los pesos y descontamos costes en los retornos *stratReturnsXX* de acuerdo con la variable kappa ver Figura 42.

```
norm11=abs(wNew1-wRearranged1);
norm12=abs(wNew2-wRearranged2);
norm13=abs(wNew3-wRearranged3);
norm14=abs(wNew4-wRearranged4);

stratReturnsSP(end)=(1+stratReturnsSP(end))*(1-kappa*sum(norm11))-1;
stratReturnsSPeuro(end)=(1+stratReturnsSPeuro(end))*(1-kappa*sum(norm12))-1;
stratReturnsSPsmall(end)=(1+stratReturnsSPsmall(end))*(1-kappa*sum(norm13))-1;
stratReturnsSPeurosmall(end)=(1+stratReturnsSPeurosmall(end))*(1-kappa*sum(norm14))-1;
```

Figura 42. Cálculo de costes de transacción.

Como vemos la comparación será entre los pesos reordenados *wRearranged* y la nueva matriz de pesos *wNew*. El coste de esta transacción se aplica al último dato de nuestros retornos y se reflejará en la web esa misma semana. Con este último cálculo dejamos las carteras preparadas, para la próxima semana continuar con el procesamiento normal.

7. Descarga y cambio de moneda de índices.

También iremos actualizando los datos de los índices con los que nos comparamos Eurostoxx y S&P 500. Este apartado es exactamente igual que el explicado en la simulación histórica.

8.Cálculo de medidas de rendimiento.

Por último tenemos que volver a calcular y escribir en ficheros todos los datos de rendimiento. Al estar operando siempre con nuestro vector de rentabilidades el código utilizado es el mismo que en la parte de simulación histórica y que explicaremos más adelante ver Figura 43.

```
weeklyHistoTickers;
chartsAndData;
```

Figura 43. Rutinas de cálculo de rendimiento semanales.

La función *chartAndData.m* será explicada en detalle más adelante en el apartado 3.2. La otra función se encarga de concatenar adecuadamente las matrices históricas de tickers y pesos. Que posteriormente serán volcados en sus respectivos ficheros.

9.Tratamiento de errores.

Si se produce un error durante la ejecución no realizaremos ningún cambio en la web, se mantendrá la carpeta de la semana pasada y se destruirá en la nueva dónde estábamos trabajando. Además se enviará un correo de notificación a través de una cuenta de correo electrónico en *gmail* con una descripción del error a las

direcciones especificadas. Para ello todo el código comentado hasta ahora en las funciones *weeklyEU.m* y *weeklySP.m* está contenido en un estructura *try catch* que controla los posibles problemas.

3.1.3 Estrategias y ajuste de parámetros.

Las estrategias usadas para la predicción de carteras han sido suministradas por los tutores de este proyecto y se puede encontrar información detallada en la bibliografía adjunta [25][26][27][28]. No obstante en este apartado se dará una breve explicación sobre su funcionamiento.

Dada una matriz de mercado, se trata de encontrar la cartera de inversión que minimice la variabilidad de la rentabilidad de la cartera. Si llamamos R_t a la rentabilidad de N activos x' a nuestra cartera y Σ a nuestro estimador. Intentaremos resolver el siguiente problema con la restricción de que la cartear sea mayor o igual que cero.

$$x'\Sigma x$$
$$x'i = 1$$
$$x \geq 0$$

El estimador muestral de la matriz de varianzas y covarianzas se define de la siguiente manera:

$$\bar{\Sigma} = \frac{\sum_{i=1}^{T}(R_t - \bar{R})^2}{T-1}$$

Nunca sabremos el estimador exacto de la cartera, y el estimador calculado tendrá siempre cierto error. Intentaremos disminuir el error cuadrático medio del estimador muestral obtenido y para ello se utilizan métodos de encogimiento o de *shrinkage*. A partir del estimador muestral anterior se calcula el estimador encogido o tal y como viene en la bibliografía *shrinkage estimator*:

$$\rho\hat{\Sigma}_{NxN} + (1-\rho)Target_{NxN}$$

El parámetro ρ se elige para minimizar el error y el Target será el parámetro con el que jugaremos nosotros. Las 3 estrategias que usamos en este proyecto no es otra cosa que distintos target para el estimador encogido.

Volviendo a Matlab las 3 rutinas de *shrinkage* o encogimiento son *ShrinkEqCorr.m ,ShrinkFactor1.m, ShrinkMidentity.m*. Y la forma en que usaremos estas rutinas es la siguiente ver Figura 44.

```
[SigmaEqCorr]=ShrinkEqCorr(training);
[w1, exitflag1] = LowVolPortfolio(SigmaEqCorr,cutoff,kmax1);

[SigmaFactor1]=ShrinkFactor1(training);
[w2, exitflag2] = LowVolPortfolio(SigmaFactor1,cutoff,kmax2);

[SigmaMidentity]=ShrinkMidentity(training);
[w3, exitflag3] = LowVolPortfolio(SigmaMidentity,cutoff,kmax3);
```

Figura 44. Diferentes estrategias utilizadas.

El parámetro de entrada de estas rutinas es la matriz de mercado en forma de retornos y la salida será una nueva matriz con el estimador cambiado, que nos servirá como entrada a la rutina de generación de portfolios *LowVolPortfolio.m.* Los pesos generados en este caso serían *w1*, *w2* y *w3* para cada una de nuestras estrategias.

En principio no sabemos cuál es la mejor estrategia a la larga y será necesario crear unas rutinas que nos ayuden a su elección. Estas rutinas son independientes y no afectan ni en el servidor ni en las rutinas semanales de Matlab o de simulación histórica. Estas rutinas nos fijarán 3 variables que no se modificarán a lo largo de toda la simulación y actualización semanal. Es posible que a posteriori debido al comportamiento variable de los mercados se puedan modificar si mejoran sustancialmente los rendimientos de nuestras carteras, pero no se comprobará dinámicamente.

Los datos que vamos a obtener son 3:

- Estrategia óptima de las tres proporcionadas.
- Ventana de estimación óptima, esto es el número de días cotizados óptimo que utilizaremos para entrenar nuestras estrategias.
- Y por último el tiempo de rebalanceo entre carteras óptimo.

Cuando se habla de óptimo nos referimos siempre a mejorar el *sharpe ratio,* de nuestras inversiones. Hay que tener en cuenta que lo que es óptimo hoy puede no serlo mañana ni lo fue siempre en el pasado, los datos aquí obtenidos no son absolutos sino orientativos y relativos a la fecha en que se ejecutaron las rutinas.

Usaremos dos scripts que nos permitirán obtener unas gráficas de la evolución del *sharpe ratio,* en función de la ventana de estimación o del periodo de rebalanceo. En ellas dibujaremos cada una de las tres estrategias proporcionadas.

Estimación de ventana y *sharpe ratio*

El script utilizado es *windowEstimation.m.* Aplicaremos las 3 estrategias que tenemos a las matrices de mercado del Eurostoxx y del S&P 500. Simularemos históricamente de la misma forma que ya explicamos en las rutinas de simulación

histórica para cada una de los tamaños de ventana y calcularemos el *sharpe ratio* de la siguiente manera:

```
%mean
means=252*squeeze(mean(stratGains(end-490:end,:)));
%standard deviation
stdevs=sqrt(252)*squeeze(std(stratGains(end-490:end,:),0,1));
%sharpe ratio
sharRat(i,:)=means./stdevs;
```

Figura 45. Cálculo del Sharpe Ratio.

La primera operación es la media anualizada de los rentabilidades o retornos obtenidos. Teniendo en cuenta que un año tiene alrededor de 252 días de cotización calcularemos la media con la función de Matlab *mean.m* y multiplicaremos por los días de cotización en un año:

$$mean = 252 * \sum_{i=1}^{n} \frac{Rentabilidad_i}{n}$$

Para la desviación estándar usaremos la función de Matlab *std.m*, de nuevo también la anualizaremos:

$$stdevs = \sqrt{252} \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(Rentabilidad_i - \overline{Rentabilidad}\right)^2}$$

Finalmente el *sharpe Ratio* será la relación entre estas dos medidas, aumentando si aumenta nuestra rentabilidad pero disminuyendo si lo hace la varianza o volatilidad del mercado. Podemos decir que mide el exceso de rendimiento por unidad de riesgo en una inversión [30].

Para este apartado fijaremos el periodo de rebalanceo en un valor fijo de 40 días. Las gráficas obtenidas entre el 14 y 15 de Octubre de 2013 para el Eurostoxx y el S&P 500 son las siguientes Figura 46 y Figura 47:

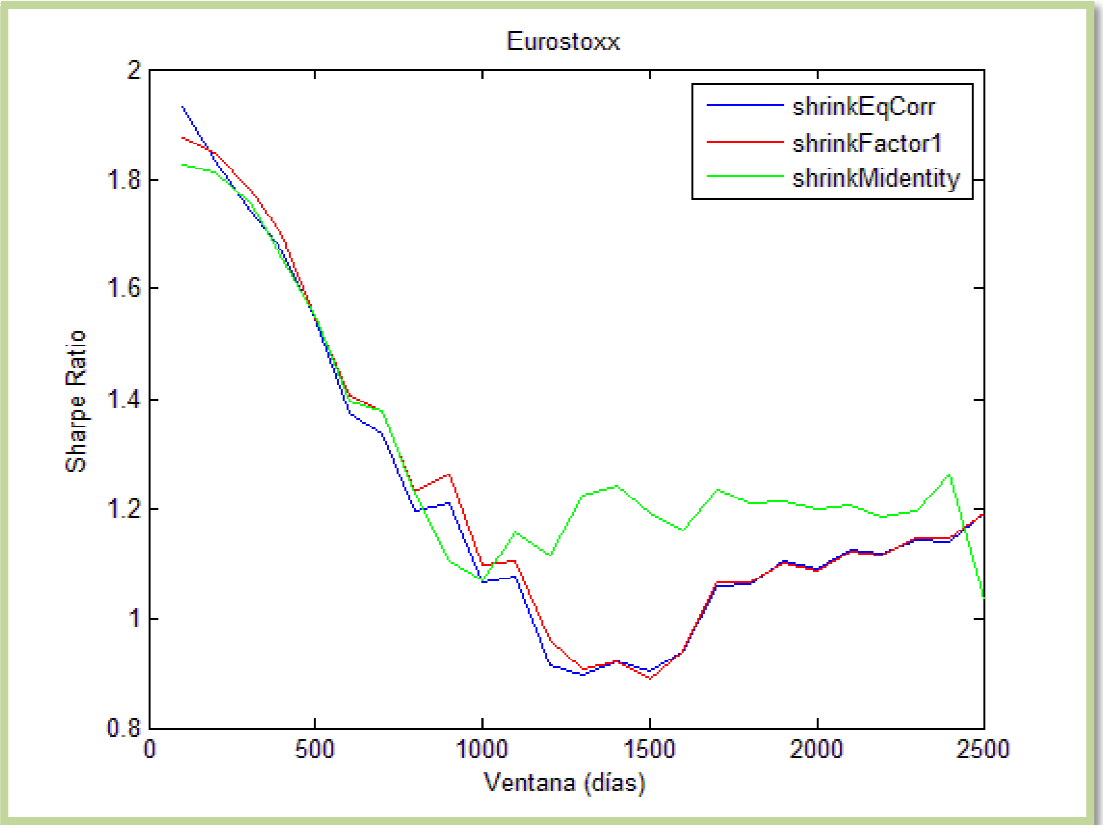

Figura 47. Gráfica del Sharpe Ratio en función de la ventana, Eurostoxx.

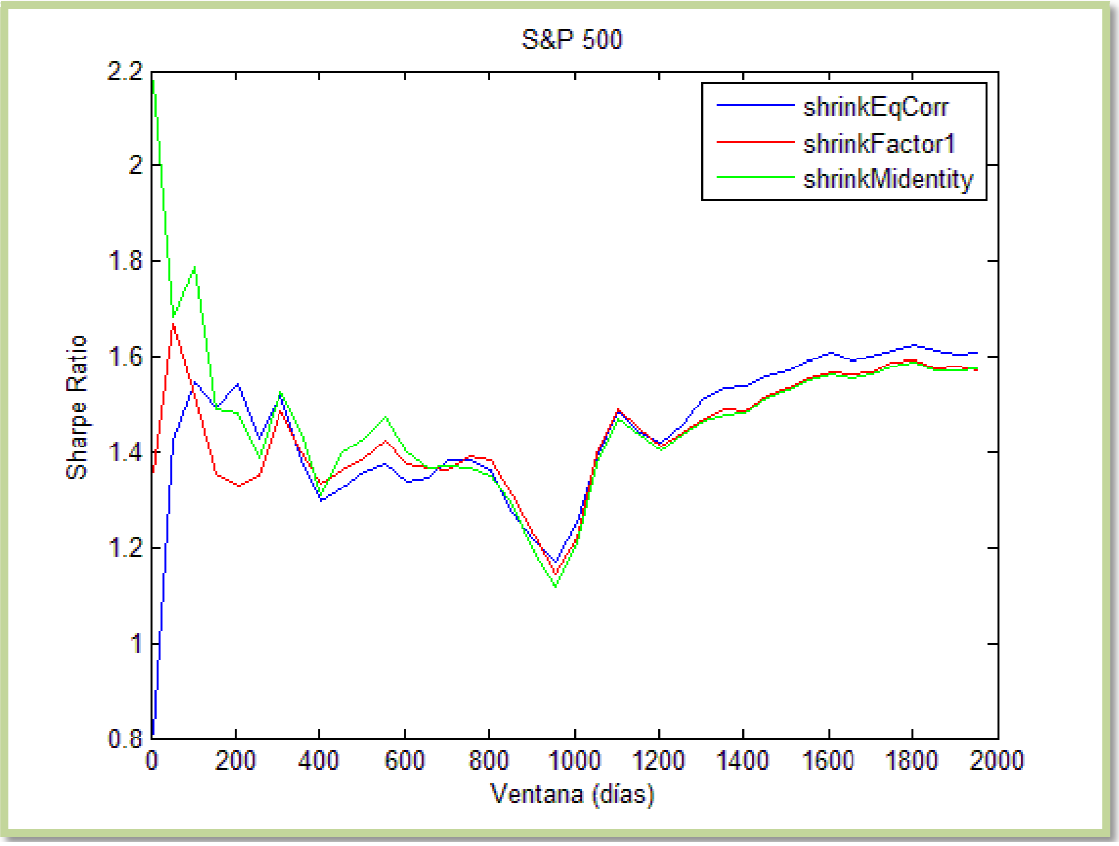

Figura 46. Gráfica del Sharpe Ratio en función de la ventana, S&P500.

Observando ambas figuras escogimos una ventana de estimación de 1700 días cotizados, ya que alrededor de esta cifra las estrategias parecen más estables. En cuanto a la estrategia a utilizar mirando la gráfica del mercado S&P 500 no hay gran diferencia entre las 3 estrategias y sin embargo observando el comportamiento en el Eurostoxx podemos ver que la estrategia *shrinkMidentity* es superior durante una gran parte de la simulación.

Es sencillo demostrar la variabilidad de estas gráficas en función de cómo se comporte el mercado, para ello haremos otra simulación retrasando los datos 500 días, por tanto el último dato que tendremos en nuestra matriz de mercado estará ubicado en el mes de Octubre de 2011, recordemos que 500 días cotizados equivalen a unos 2 años reales ver Figura 48.

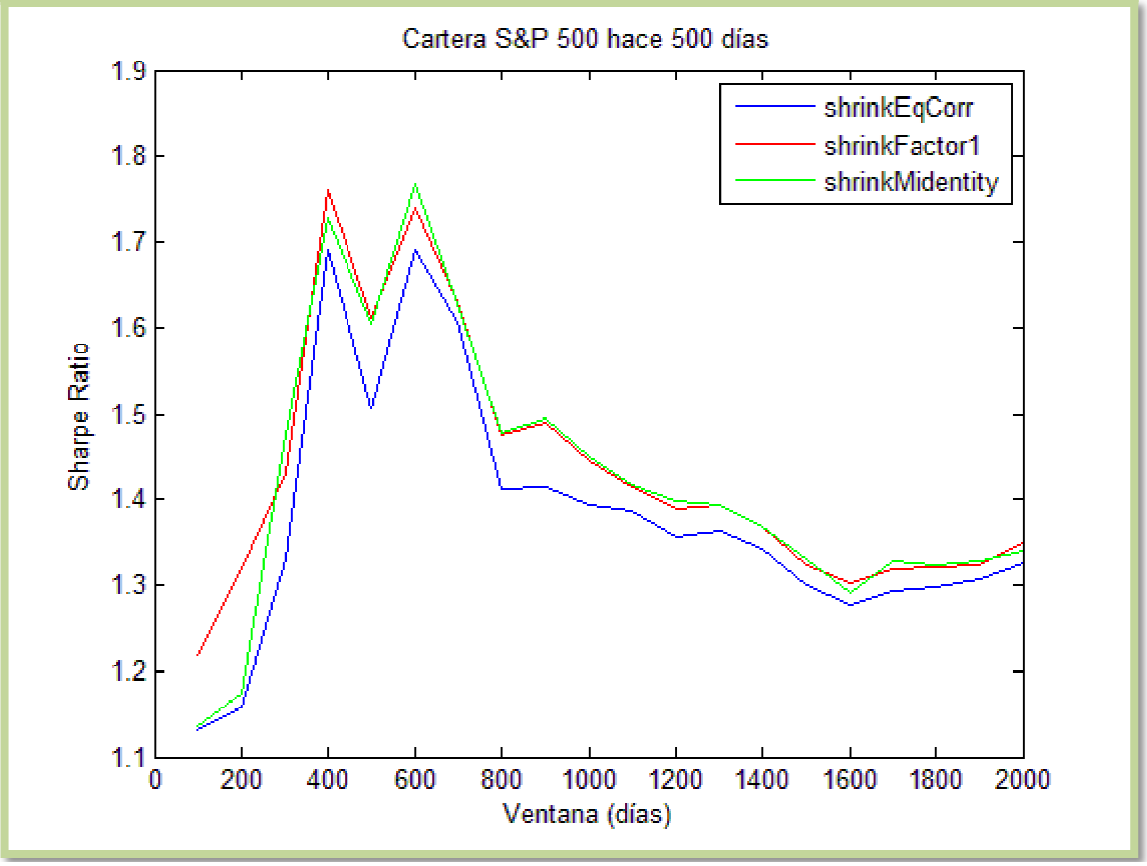

Figura 48. Gráfica del Sharpe Ratio en función de la ventana, S&P 500. Hace 500 días.

Podemos apreciar cómo para ventanas pequeñas el *sharpe ratio* es muy variable y cómo de nuevo se estabiliza para tamaños de ventana de 1700 días. Si nos fijamos detenidamente la depresión que sufríamos antes en la figura del S&P 500 se ha movido de la posición 1000 a la posición 500. Esto nos indica que el rendimiento de la ventana depende también del estado del mercado en cada momento. Mirando las estrategias *shrinkEqCorr* es inferior a las otras dos, que entre si son muy similares.

Periodo de rebalanceo

En este paso fijaremos el tamaño de la ventana en 1700 y trataremos de seleccionar el periodo de rebalanceo más conveniente. Los datos obtenidos para Eurostoxx y S&P 500 son muy similares, mostraremos únicamente los obtenidos en la cartera del S&P 500, ver Figura 49.

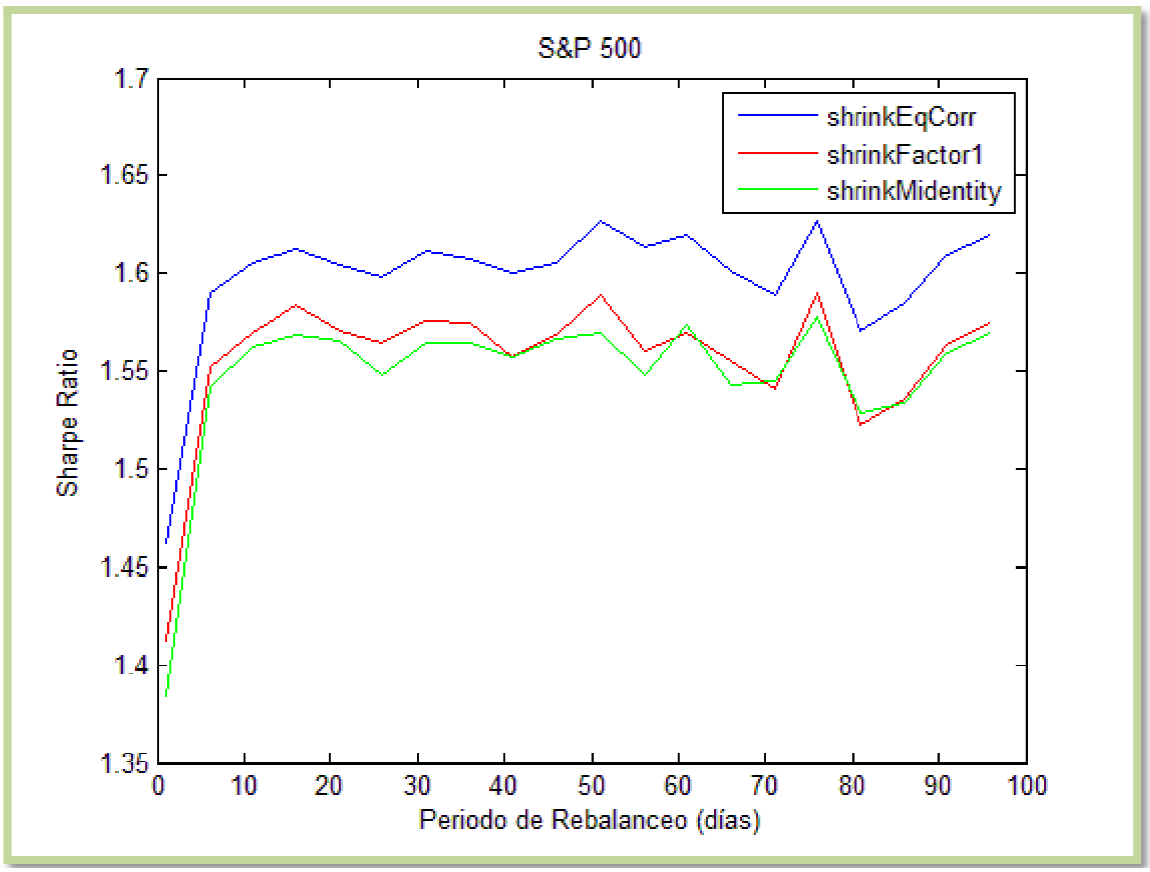

Figura 49. Gráfica del Sharpe Ratio en función del periodo de rebalanceo, S&P500.

Podemos sacar dos conclusiones, la primera es que rebalanceos cortos nos reducen el *sharpe ratio*, esto se debe a que al cambiar la cartera tenemos unas pérdidas debidas a los costes de transacción de 40 puntos básicos y si cambiamos muy frecuentemente perderemos rentabilidad. La segunda conclusión es que a medida que aumentamos el periodo el *sharpe ratio* se hace inestable. Esto también es lógico pues si aplicamos estrategias cada periodos muy largos nuestra predicción de cartera óptima perderá fiabilidad a lo largo del tiempo y observaremos una aleatorización respecto a la rentabilidad y volatilidad obtenidas.

Se elegirá un periodo de rebalanceo de 40 días cotizados lo que equivale a cambiar nuestra cartera cada dos meses, un periodo razonable de cara a posibles clientes y dentro de la zona estable como veíamos en la figura.

3.2 Datos ofrecidos en la página web.

En este apartado trataremos de detallar todos los datos ofrecidos en la plataforma web y que han sido generados por Matlab. Se mostrarán algunas de las operaciones y funciones utilizadas para calcular estos datos así como su formato en los ficheros, también se mostrarán y se explicarán las gráficas ofrecidas en la página web y generadas por Matlab.

Podemos distinguir 3 grandes apartados:

- Históricos de carteras.
- Medidas de rendimiento.
- Gráficas de rentabilidad y riesgo.

El script de Matlab encargado de generar todos estos datos se llama *chartsAndData.m* y es común tanto para las rutinas de simulación histórica como para las rutinas semanales.

3.2.1 Históricos de carteras.

Todos los datos aquí calculados serán mostrados en la pestaña web *Portfolio*. En esta pestaña ofreceremos al usuario todas las carteras calculadas a lo largo de la historia con el nombre de los tickers de las empresas y su correspondientes pesos. Para poder ofrecer nuestras carteras históricas necesitaremos guardar antes 3 grupos de datos ver Figura 50:

- Fechas: las rentabilidades y carteras calculadas no son continuas en el tiempo y dependen de los días de cotización, es necesario ofrecer a posibles usuarios interesados las fechas exactas de nuestras carteras.
- Tickers: son los nombres que usaremos para identificar a las empresas o activos. La matriz de empresas irá cambiando con el tiempo, siendo necesario su almacenamiento diario.
- Pesos: porcentaje de inversión en cada activo, almacenado diariamente en paralelo con los tickers correspondientes.

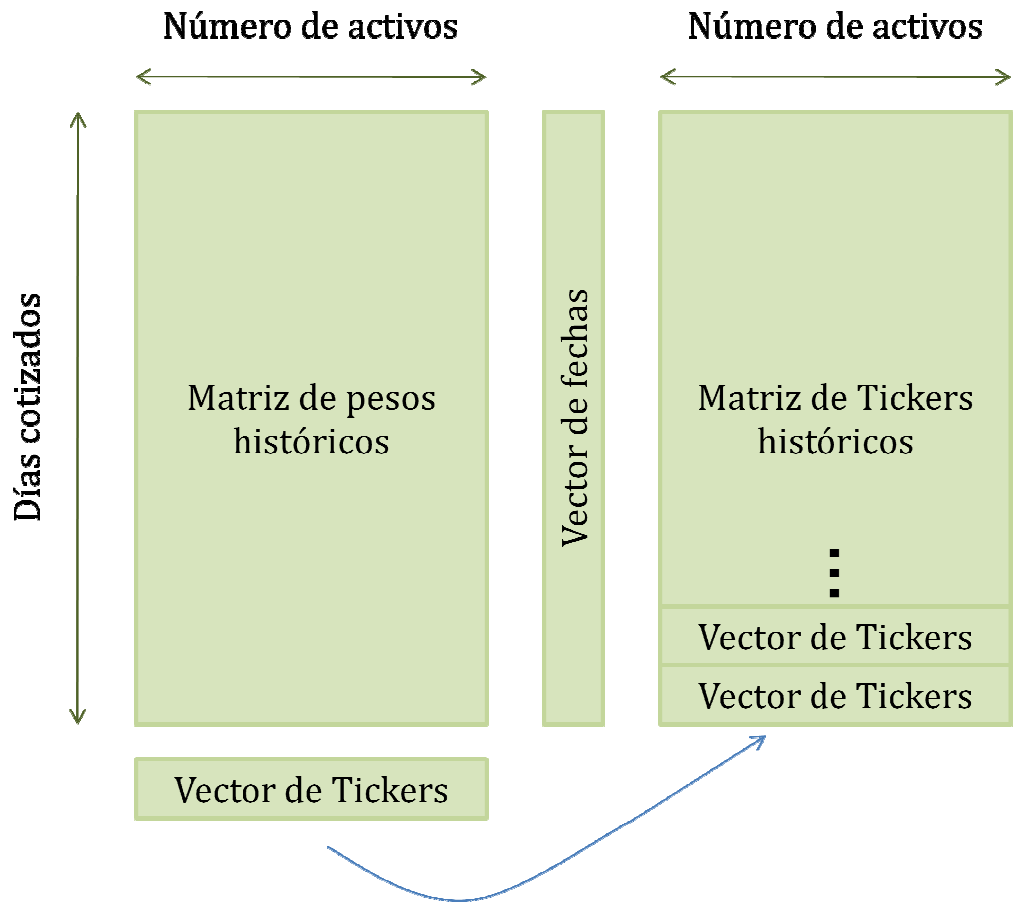

Figura 50. Esquema de las matrices históricas.

Fechas

El vector de fechas guardado es la variable *xdates* en Matlab, si bien durante toda la ejecución pasaban dos cosas, las fechas están guardadas en formato numérico y la posición uno del array correspondía siempre a la fechas más antigua, a la hora de guardar se realizará a la inversa para facilitar las cosas en la presentación web y además se escribirán en formato día-mes-año (dd-mmm-yyyy), ver Figura 51.

```
%guardamos el fichero de fechas
datesString=datestr(xdates(2:end),'dd-mmm-yyyy');
datesStringReversed=flipud(datesString);
```

Figura 51. Formato de ficheros de fechas.

Además todos los archivos serán guardados en formato *.csv*. En el caso del fichero de fechas sólo tiene una columna y tiene el siguiente formato ver Figura 52.

Figura 52. Fichero de fechas.

Cada fila del archivo de fechas coincide con cada fila de los archivos de pesos y tickers. Sólo se tendrá un archivo de fechas, que comparten todas las carteras.

Pesos y Tickers históricos

Tendremos 4 matrices de pesos y 4 de tickers para cada mercado, representarán el porcentaje de inversión en cada activo y los tickers correspondientes. En el caso de S&P 500 tendremos unas matrices de unos 450 activos y cada día dependiendo de la cartera sólo tendremos entre 6 y 25 empresas en las que invertiremos. No será necesario por tanto guardar todos los pesos y tickers, sólo aquellos cuyo valor sea distinto de cero, esto también facilitará su lectura en la plataforma web. Debido a las operaciones realizadas con Matlab cada activo tiene un número variable de decimales, que también será regulado de cara a la presentación web.

La representación de los pesos estará en porcentaje y con dos decimales como máximo por activo, y el archivo de nuevo tendrá formato *.csv* delimitado por comas ver Figura 53.

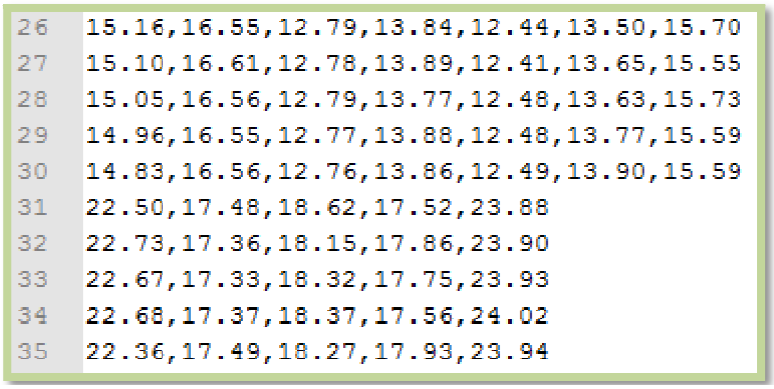

Figura 53. Fichero de pesos.

La correlación de estos pesos tiene que ser uno a uno con la matriz de tickers, por ello y aunque el vector de tickers completo es común para las 4 carteras a lo largo de la ejecución, ahora sólo queremos guardar los tickers de las empresas en los que invertimos cada día. El fichero correspondiente al caso que veíamos en la figura 53

para Eurostoxx cartera pequeña y en euros, se corresponde con el de la figura 54 siguiente.

```
26  ABI.BR,BN.PA,DTE.DE,EI.PA,OR.PA,ORA.PA,SAP.DE
27  ABI.BR,BN.PA,DTE.DE,EI.PA,OR.PA,ORA.PA,SAP.DE
28  ABI.BR,BN.PA,DTE.DE,EI.PA,OR.PA,ORA.PA,SAP.DE
29  ABI.BR,BN.PA,DTE.DE,EI.PA,OR.PA,ORA.PA,SAP.DE
30  ABI.BR,BN.PA,DTE.DE,EI.PA,OR.PA,ORA.PA,SAP.DE
31  ABI.BR,DTE.DE,OR.PA,ORA.PA,SAP.DE
32  ABI.BR,DTE.DE,OR.PA,ORA.PA,SAP.DE
33  ABI.BR,DTE.DE,OR.PA,ORA.PA,SAP.DE
34  ABI.BR,DTE.DE,OR.PA,ORA.PA,SAP.DE
35  ABI.BR,DTE.DE,OR.PA,ORA.PA,SAP.DE
```

Figura 54. Fichero de Tickers.

El cambio en el número de activos entre la línea 30 y 31 se corresponde a un rebalanceo de cartera.

Todos estos ficheros serán leídos por el servidor mediante unas librerías preparadas para leer archivos en este formato *.csv*.

3.2.2 Medidas de rendimiento.

Todas nuestras carteras recomendadas tendrán unos datos de interés que nos indican cómo se están comportando. Estos datos también serán ofrecidos en la página web y serán generados con Matlab. Podremos encontrarlos en la pestaña *Performance Measures* de la web y aquí explicaremos cuales son cómo se calculan y el formato de guardado:

- Sharpe Ratio (ratio de Sharpe).
- Volatilidad anualizada.
- Valor en riesgo.
- Máxima caída.
- Retornos acumulados.
- Medias de retorno anualizadas.

Cada una de estas medidas será calculada hasta 12 veces, 4 para las carteras europeas, 4 para las carteras americanas, 2 para el índice europeo una en cada moneda y otras 2 para el S&P 500 una en cada moneda. Las 6 correspondientes a cada mercado se escribirán en un fichero ubicado en la carpeta correspondiente a su mercado. Por tanto generaremos dos ficheros de 6 datos cada uno, para cada una de estas medidas. En algunos casos nos interesará calcular la medida para diferentes periodos. En el caso del *sharpe ratio* generaremos ficheros para 1, 3 y 5 años y para cada mercado, es decir, un total de 6 ficheros, 3 europeos y 3 americanos.

Todas estas medidas seguirán el siguiente formato para su posterior presentación en la página web ver Tabla 5.

Moneda	Cartera pequeña	Cartera grande	Eurostoxx
Euros	1.02	0.99	0.5
Dólares	0.83	0.86	0.48

Tabla 5. Formato medidas de rendimiento.

Este ejemplo corresponde concretamente al fichero europeo donde guardamos las medidas correspondientes al *sharpe ratio* de los últimos 5 años. Y el formato que seguirán todas estas tablas en los ficheros será, un .csv delimitado por comas ver Figura 55.

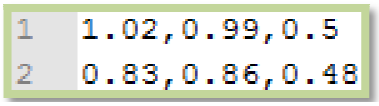

Figura 55. Fichero medidas de rendimiento.

Para escribirlos en estos ficheros usaremos la función de Matlab *csvwrite(FILENAME,Data)*.

Sharpe ratio

Hasta ahora el *sharpe ratio* es la única medida que hemos estado nombrando y debido a su relevancia a la hora de construir el sistema ya ha sido explicada en el apartado de estrategias, así como los cálculos intermedios de la volatilidad y la media. Los conceptos y cálculos realizados son los mismos que han sido explicados en el apartado 3.1.3 por lo que no los repetiremos aquí.

El único aspecto a resaltar es que se calculará el *sharpe ratio* de los últimos 5 años de los últimos 3 años y del último año mediante Matlab. Generaremos un fichero distinto para cada uno de estos periodos con el formato ya comentado. En total 6 ficheros, 3 del Eurostoxx y 3 del S&P 500.

Se puede comprobar que el *sharpe ratio* coincide de la división de la media anualizada y la volatilidad anualizada.

Volatilidad anualizada

La volatilidad es un valor que nos ofrece información de la frecuencia y de la intensidad con la que se producen cambios en nuestras carteras de inversión, por tanto nos interesará que este valor sea bajo.
Al ser la volatilidad una medida necesaria para calcular el *sharpe ratio* también se explicó cómo obtenerla en el apartado 3.1.3.

Al igual que el *sharpe ratio* calcularemos la volatilidad de los 5 últimos años, de los 3 últimos y del último año. En total 6 ficheros, 3 del Eurostoxx y 3 del S&P 500.

Valor en riesgo

Esta medida se usa en matemáticas financieras para medir el riesgo. A veces se abrevia *VaR* a partir de su expresión al inglés *Value at Risk* [31].
El valor en riesgo mide la cantidad máxima que podemos perder en nuestra cartera de inversión con una determinada probabilidad. Se calculan dos valores típicos usados en finanzas, el *VaR* con 1% y 5% de probabilidad. Por tanto obtendremos la máxima perdida de nuestras carteras con un 99% y un 95% de probabilidad. Interesará tener un valor en riesgo pequeño.

Para hallar esta medida nos apoyaremos en una función de Matlab que calcula percentiles *prctile.m* ver Figura 56.

```
VaRIDef1Y1 = -prctile(returnsI(end-251:end),1,1);
```

Figura 56. Rutina de cálculo del VaR.

Básicamente ordena los datos y nos dice por debajo de que valor se encuentra un porcentaje dado de observaciones.

Al igual que el *sharpe ratio* calcularemos la volatilidad de los 5 últimos años, de los 3 últimos y del último año. Cómo calculamos además VaR para 1% y 5% en este caso tendremos en total 12 ficheros, 6 del Eurostoxx y 6 del S&P 500.

Máxima caída

En inglés *maximum drawdown*, se define como la máxima caída experimentada por una inversión en un periodo determinado. Interesará que sea pequeño.

Para calcularlo al igual que el valor en riesgo nos apoyaremos en una función ya existente en Matlab *maxdrawdown(Data).*

Al igual que el *sharpe ratio* calcularemos la máxima caída de los 5 últimos años, de los 3 últimos y del último año. En total 6 ficheros, 3 del Eurostoxx y 3 del S&P 500.

Retornos acumulados

Se mostrarán también los retornos acumulados en el último año, el último trimestre y el último mes. Usaremos periodos más pequeños de estos retornos para ofrecer una rápida lectura que nos aporta información cuantitativa de nuestras inversiones en cortos periodos de tiempo.

Como tenemos el histórico de rentabilidades en un vector de retornos, para calcular esta acumulación usamos la función de Matlab que convierte de retornos a precios *ret2price.m.*

```
acumIYTDdolar=ret2price(returnsI(end-251:end,:),[],[],[],'Periodic');
```

Figura 57. Cálculo de retornos a partir de precios.

Generaremos un fichero distinto para cada uno de los periodos mencionados con el formato ya comentado. En total 6 ficheros, 3 del Eurostoxx y 3 del S&P 500.

Media anualizada

Media de los retornos que podemos esperar de nuestras carteras al cabo de un año, por tanto nos interesará que este valor sea alto.
Al ser la media una medida necesaria para calcular el *sharpe ratio* también se explicó cómo obtenerla en el apartado 3.1.3.

Al igual que el *sharpe ratio* calcularemos la media de los 5 últimos años, de los 3 últimos y del último año. En total 6 ficheros, 3 del Eurostoxx y 3 del S&P 500.

3.2.3 Gráficas de rentabilidad y riesgo.

Ofreceremos dos tipos de gráficas. Una gráfica que representará la ganancia acumulada tanto de nuestro sistema como de los índices, y otra gráfica en la que enfrentaremos la ganancia y el riesgo de inversión.

Gráfica de ganancia

Tendremos en total 4 gráficas, 2 para el Eurostoxx y otras 2 para el S&P 500. Cada una de estas dos gráficas a su vez estarán representando euros o dólares. Y en cada gráfica podremos encontrar las evoluciones de la cartera pequeña y grande así como el índice.

```
%colors see colorSpec
col=[ 0,0,1;0 1 0; 1 0 0; 1 1 0; 1 0 1; 0 1 1];
fh=figure('Name',titleProfDef,'NumberTitle','off');
stratAc=ret2price(squeeze(fig(1,:,:)),[],[],[],'Periodic');
stratAceuro=ret2price(squeeze(fig(2,:,:)),[],[],[],'Periodic');
plot(100*stratAc(:,1)-100,'b','linewidth',1.1)
hold on
plot(100*stratAc(:,2)-100,'g','linewidth',1.1)
plot(100*cumIndW(:,1)-100,'color',col(5,:),'linewidth',1.1);
title(titleProfDef);
ylabel('Cumulative Returns (%)')
xlabels=datestr([xdates(1) xdates(floor(end/3)) xdates(floor(2*end/3)) xdates(end)]);
xticks=[1 floor(length(xdates)/3) floor(2*length(xdates)/3) length(xdates)];
xlim([1 length(xdates)]);
set(gca,'XTick',xticks)
set(gca,'XTickLabel',xlabels)
set(gca,'XMinorTick','on')
legend('SmallPortfolio','LargePortfolio',legendIndex,'Location','Best')
grid
hold off
saveas(gcf,figFileWDef,'jpg');
```

Figura 58. Representación de gráfico de ganancias, código Matlab.

Para su correcta presentación en la página web definiremos diferentes parámetros en Matlab ver Figura 58.
En el eje horizontal representaremos 4 fechas en formato dd-mmm-yyyy (ej: 22-Sep-2010). Y en el eje vertical mostraremos el porcentaje de ganancia acumulado a lo largo de la historia. Partiremos de 0%, ver Figura 59.

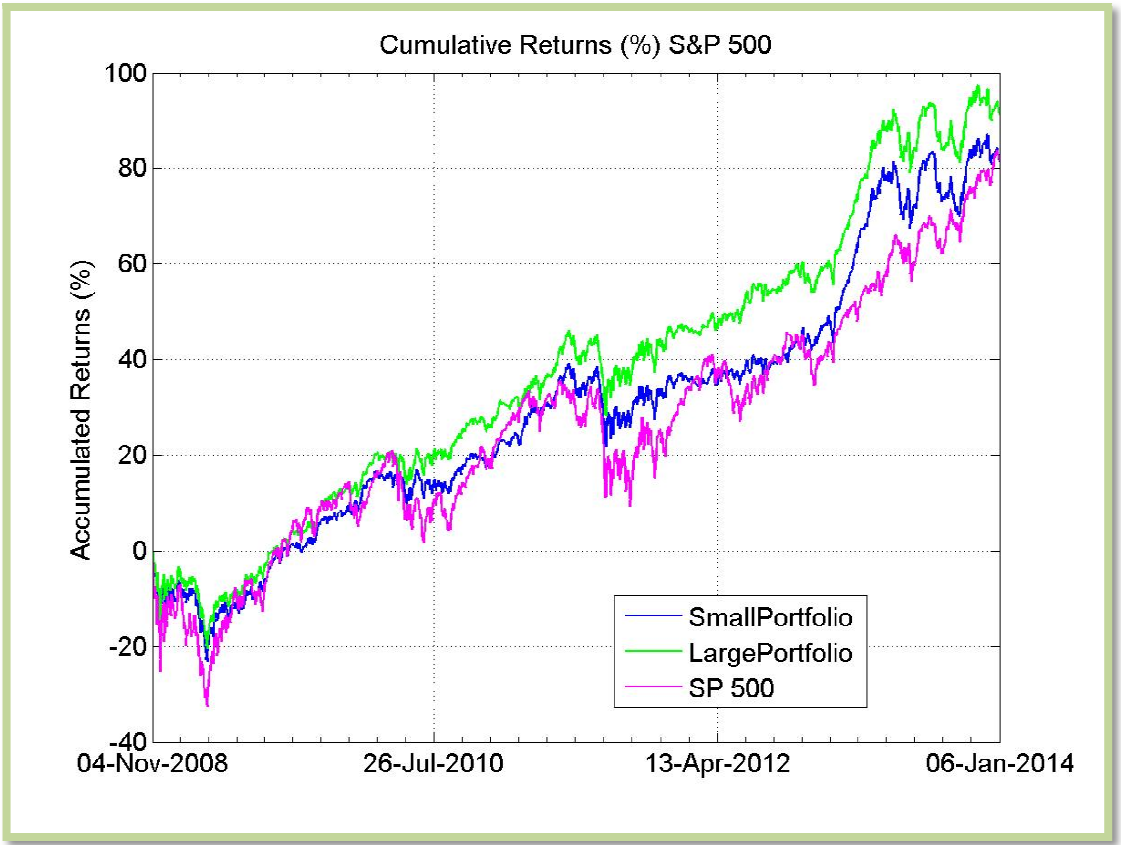

Figura 59. Gráfica de ganancia acumulada, S&P 500.

Para calcular el retorno acumulado usaremos la misma función que usábamos para calcular las medidas de rendimiento *ret2price.m*.

Gráfica de retorno vs riesgo

En este caso tendremos 6 gráficas para el Eurostoxx y 6 gráficas para el S&P 500. De estas 6 gráficas 3 representarán las carteras en euros y 3 las carteras en dólares. Y a su vez cada una de estas 3 gráficas representará un periodo distinto 1, 3 y 5 últimos años.

Para su correcta presentación en la página web definiremos diferentes parámetros en Matlab de manera similar a las gráficas de rentabilidad, ver Figura 58.

Las funciones utilizadas para calcular la media de los retornos y la volatilidad son las mismas que las explicadas en el apartado de medidas de rendimiento.

Representaremos en el eje horizontal la volatilidad asociada a cada cartera en porcentaje así como la del índice, y en el eje vertical representaremos la media anualizada de nuestras ganancias o retornos. Estos datos se calculan para periodos determinados por lo que la representación que veremos será un punto para cada cartera o índice, ver Figura 60.

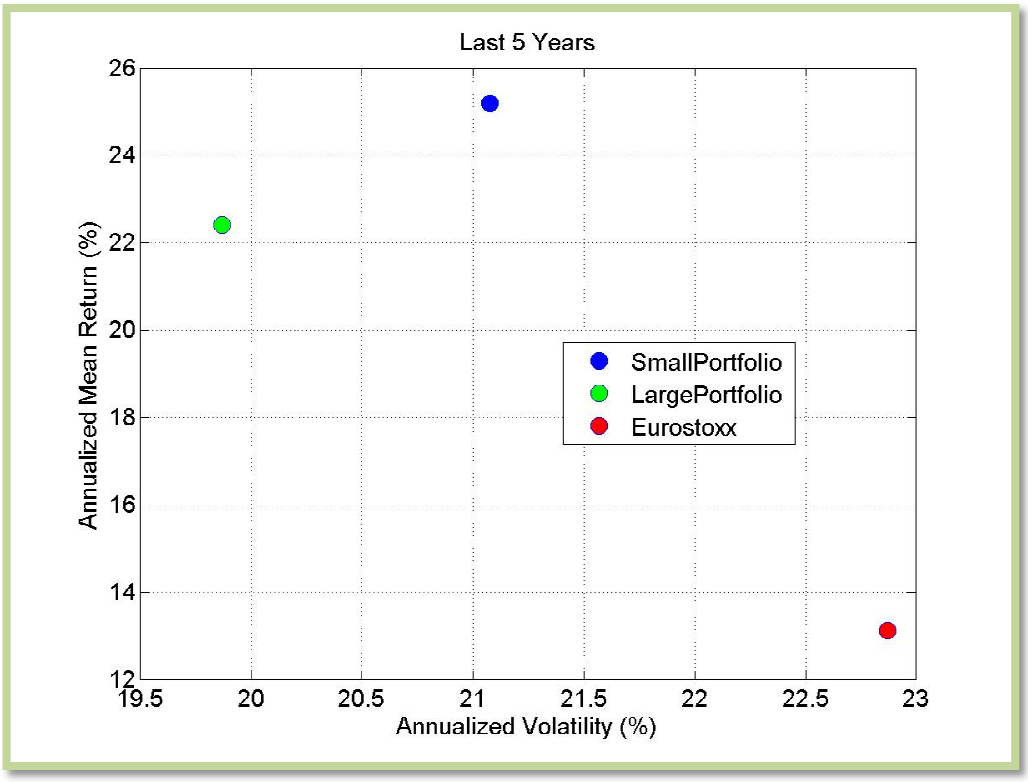

Figura 60. Gráfica de Retorno Medio vs Riesgo, S&P 500.

La situación óptima en la que queremos nuestras carteras estaría situada al noroeste del gráfico, zona que equivale a una alta rentabilidad con una menor volatilidad.

3.3 Página web

La página web permitirá el control y seguimiento de nuestras carteras ofrecidas. En ella presentaremos todos los datos hasta ahora obtenidos. El formato de los ficheros será xhtml y la tecnología que usaremos para poder ofrecer dinámicamente los datos será Java Server Faces. Todo ello irá montado sobre un servidor Apache Tomcat 7.0 instalado en Windows Server 2012 Standard, si bien se podrían utilizar otras alternativas. El entorno de desarrollo utilizado es Netbeans IDE 7.3 ya que nos permite integrar muy bien todas estas tecnologías. Finalmente se incluirán algunos javascripts de Google Analytics que nos permitirán observar la evolución del servicio web con dicha herramienta.

La página web será mostrada enteramente en inglés para poder abarcar a un mayor número de clientes, para simplificar la explicación nos referiremos a las diferente páginas creadas por su nomenclatura en inglés, aunque también haremos notar cual sería su equivalente en castellano para facilitar la comprensión.

Los archivos java serán los encargados de leer y suministrar los datos a los archivos xhtml. Para poder compartir la información lo harán a través de Beans, elementos compartidos entre estas dos tecnologías, pueden ser botones, tablas, campos de texto, etc [33].

La url utilizada durante la realización de este proyecto es http://lowvol.uc3m.es/ una vez entremos al enlace, la página web está estructurada de la siguiente manera ver Figura 61.

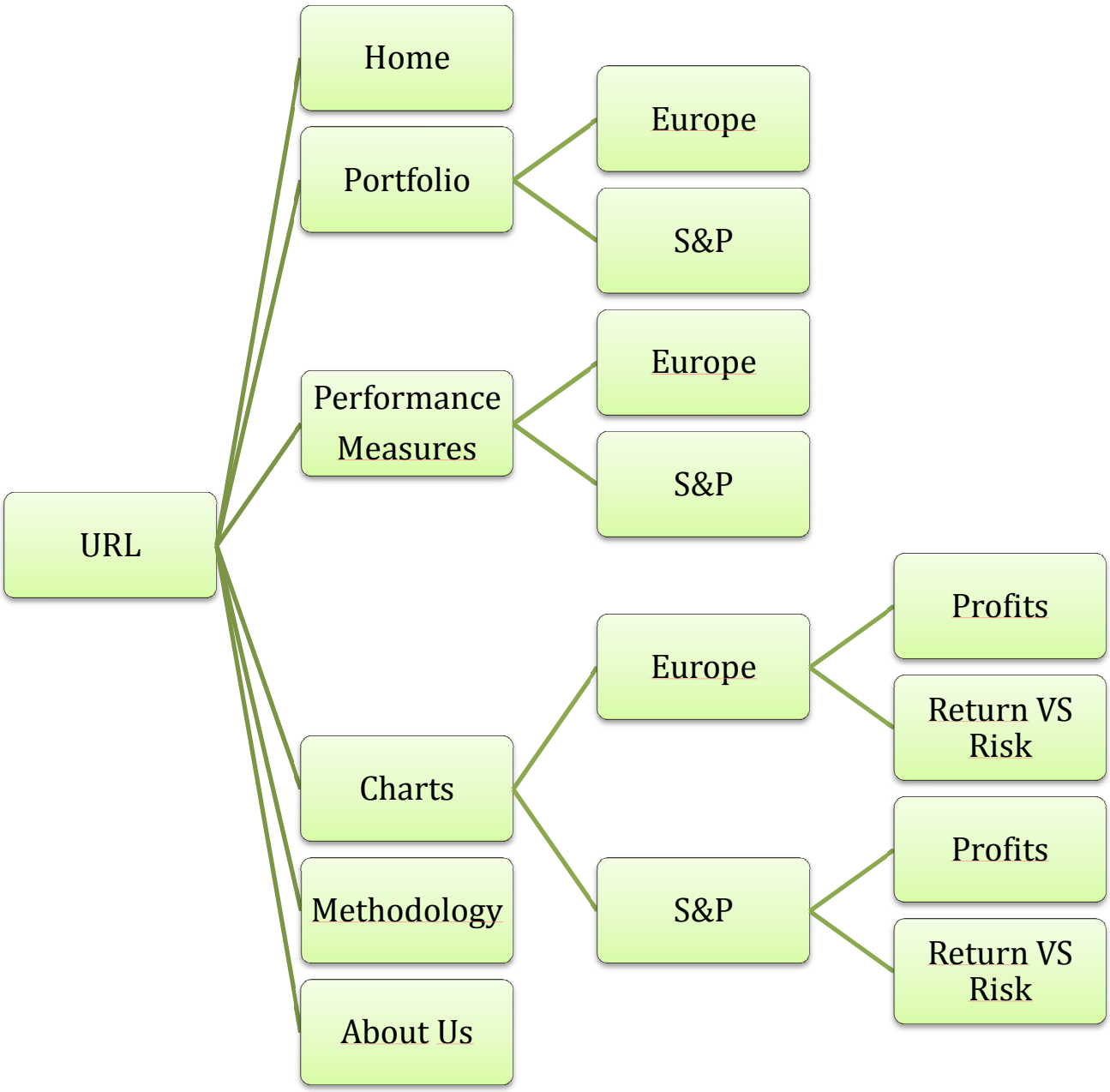

Figura 61. Esquema página Web.

Sólo los nodos finales del árbol mostrado tendrán su propio archivo xhtml que son

- *Home.*
- *Portfolio Europe.*
- *Portfolio S&P.*
- *Performance Measures Europe.*
- *Performance Measure S&P.*
- *Profits Europe.*

- *Profits S&P.*
- *Return VS Risk Europe.*
- *Return VS Risk S&P.*
- *Methodology.*
- *About Us.*

En total tendremos 11 páginas con su respectivo archivo xhtml. Y además todas las páginas web seguirán la siguiente estructura, ver Figura 62.

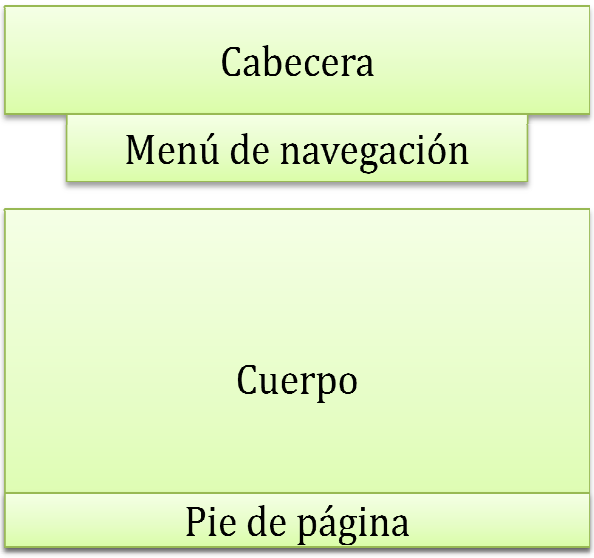

Figura 62. Disposición de la página Web.

La cabecera, el menú de navegación y el pie de página serán comunes en todas las páginas xhtml. La cabecera y el menú de navegación tienen el aspecto de la Figura 63.

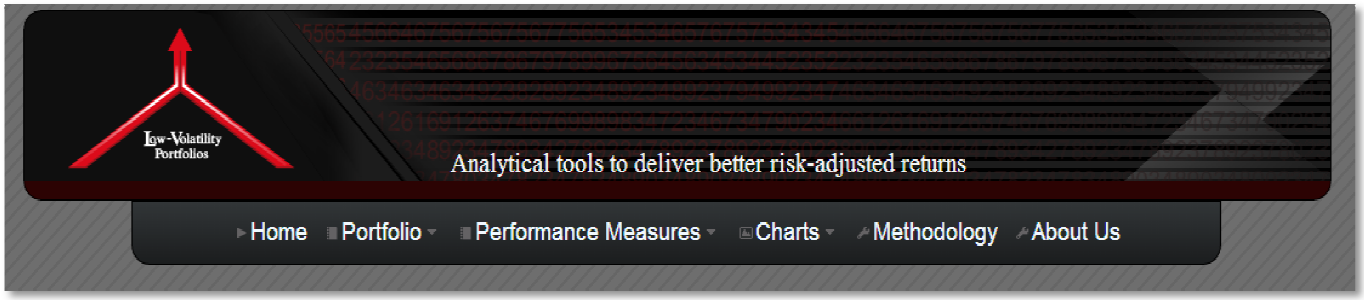

Figura 63. Cabecera de la página Web.

Con este menú desplegable podremos navegar desde cualquier página en la que estemos a cualquier otra página que queramos directamente, sin ninguna página intermedia.

Por último tendremos un archivo de estilo *.css* común a todas las páginas donde se definirán todas las especificaciones relativas a la presentación, márgenes, estilo de tablas, fuente, etc...

Página Home

Página de inicio con una breve explicación sobre lo que se ofrece y dos imágenes con gráficos de retorno vs riesgo de cada mercado enlazadas cada una con su página de origen para captar rápidamente el interés de posibles clientes, ver Figura 64.

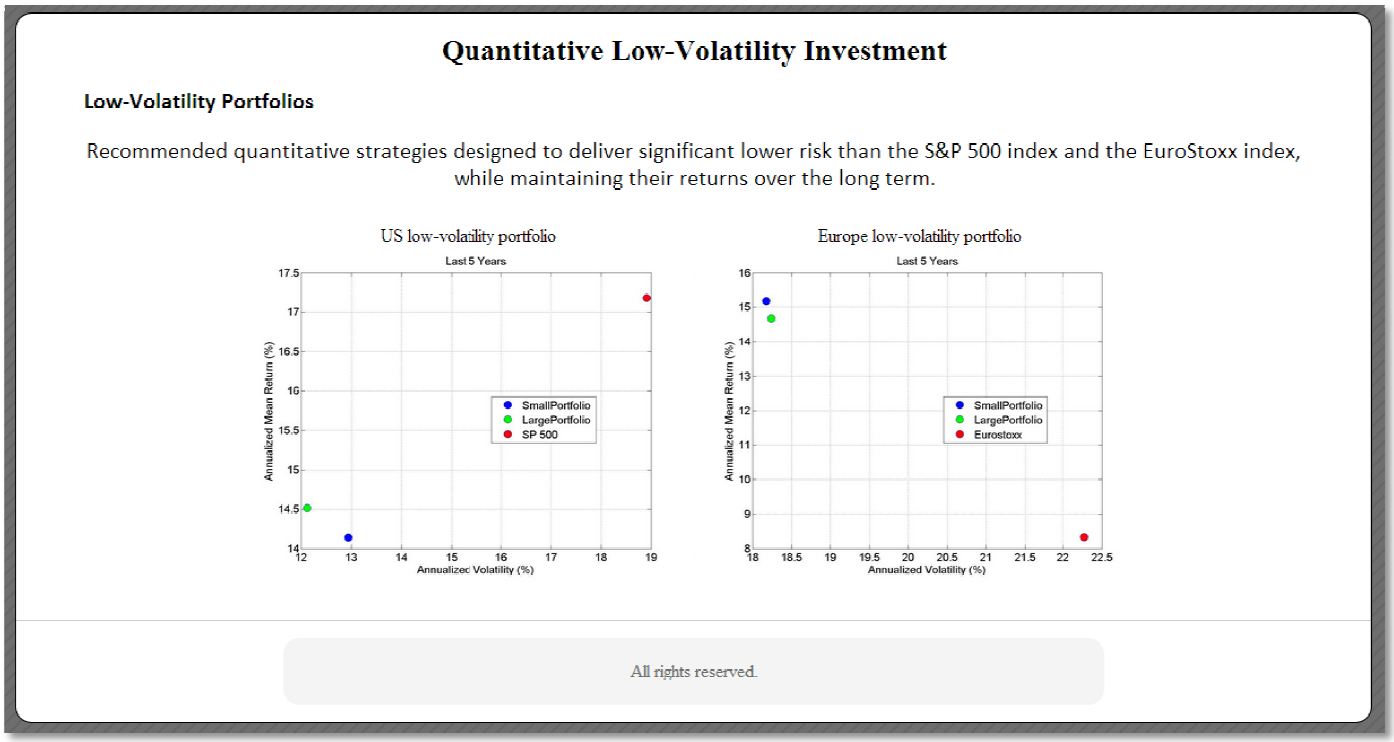

Figura 64. Página Web de inicio.

Página Portfolio

Página de carteras, tendremos una para cada mercado en ellas mostraremos 3 cosas, los tickers de las empresas a invertir, los pesos de inversión en cada una de estas empresas y un botón desplegable que permitirá al cliente seleccionar la fecha de la que quiere ver los datos. De nuevo hay que tener en cuenta que son 4 carteras por mercado:

- Cartera pequeña, en la moneda nativa del mercado.
- Cartera grande, en la moneda nativa del mercado.
- Cartera pequeña, en la moneda alternativa del mercado.
- Cartera grande, en la moneda alternativa del mercado.

En lo referente a los datos la clase de java *Portfolio.java* se encargará de leerlos y suministrarlos. Los archivos leídos son, archivo de fechas, archivos de pesos y archivos de tickers. Como tendremos dos páginas de carteras *portfolioEU.xhtml* y

portfolioSP.xhtml y ofreceremos datos diferentes en cada una de ellas para evitar duplicar código se resuelve este problema usando las propiedades de herencia de java con dos clases que heredan directamente de *Portfolio.java* que son *PortfolioSP.java* y *PortfolioEU.java* y simplemente especifican los datos particulares a cada mercado, como por ejemplo las rutas de los archivos.

La página web se muestra en la Figura 65.

Figura 65. Página Web, pestaña de carteras, Eurostoxx.

Ofrecemos también al cliente un acceso rápido a la página de medidas de rendimiento para poder ver la efectividad de estas carteras, primero ofrecemos las carteras pequeñas y después las grandes.

Página Performance Measures

Página de medidas de rendimiento, donde de acuerdo a lo explicado en el apartado 3.2.2 expondremos los datos obtenidos, respetando la estructura de los archivos. De nuevo tendremos 4 carteras para cada mercado dos archivos xhtml para cada uno y dos clases java respectivamente, que heredarán de la principal *TableBeanOne.java,* para no duplicar código. Leeremos 3 archivos por cada medida presentada correspondientes a diferentes periodos

La página web de medidas de rendimiento la mostramos en la figura 66.

Figura 66. Página Web, pestaña de medidas de rendimiento, Eurostoxx.

Para facilitar su lectura las medidas se organizarán en 3 pestañas de acuerdo con los distintos periodos y en cada una de ellas se mostrarán los datos de las 4 carteras nuestras así cómo los datos del índice correspondiente.

Las medidas serán mostradas en el siguiente orden:

- Sharpe ratio 5,3,1 últimos años.
- Volatilidad anualizada 5,3,1 últimos años.
- Valor de riesgo 99% 5,3,1 últimos años.
- Valor de riesgo 95% 5,3,1 últimos años.
- Máxima caída 5,3,1 últimos años.
- Retornos acumulados último año, trimestre y mes
- Media anualizada 5,3,1 últimos años.

La idea con este orden de presentación es presentar las medidas más atractivas primero, que son las que defiende nuestro sistema y por tanto más fuertes respecto al índice. Son aquellas referentes a la volatilidad y a largo plazo.

Página Profits

En esta página representaremos los gráficos calculados en Matlab y explicados en el apartado 3.2.3 asociados a la rentabilidad. En este caso tendremos 2 gráficos, en ellos 2 carteras serán representadas en el gráfico de dólares y las otras 2 en el gráfico de euros respectivamente, ver Figura 67.

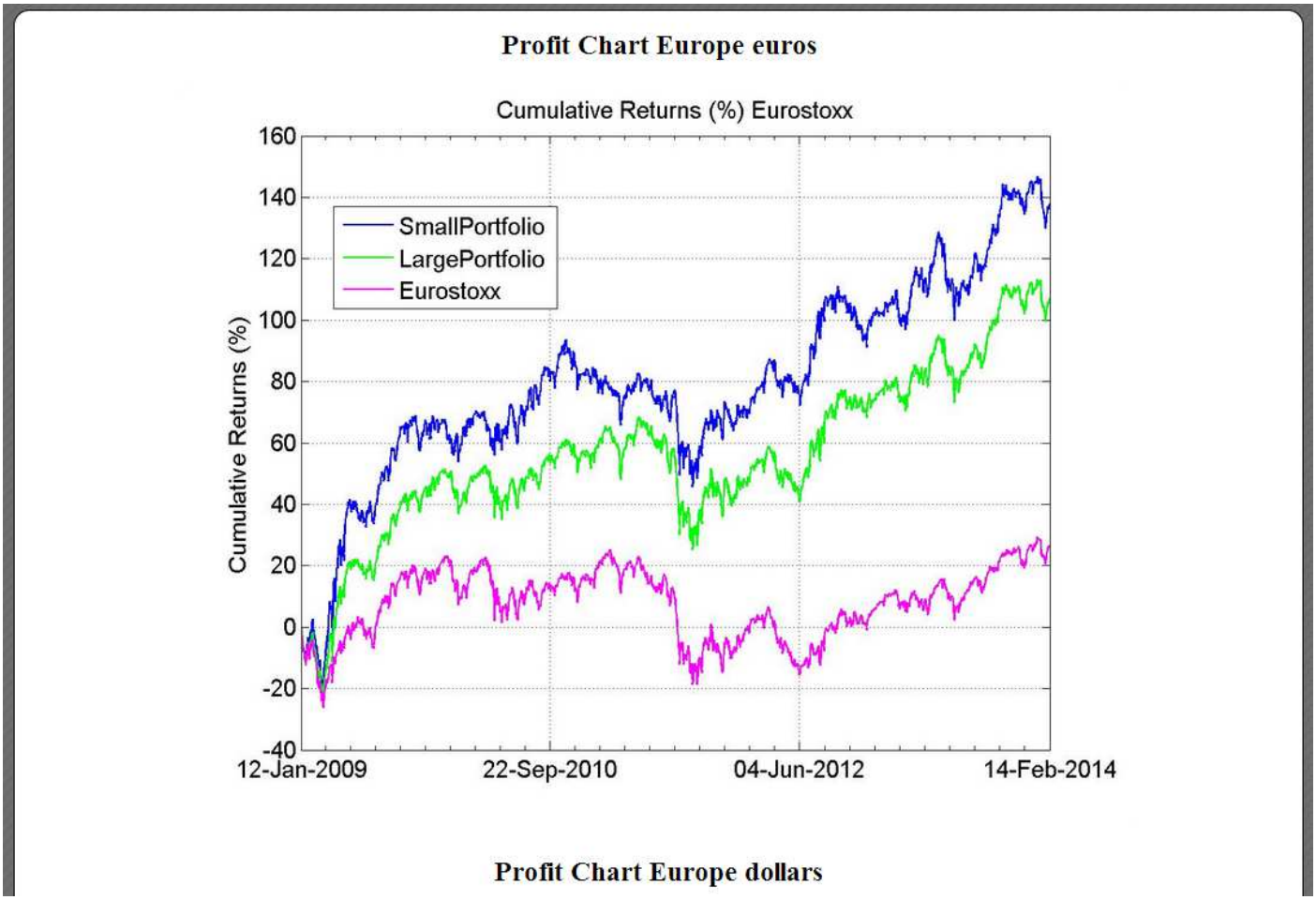

Figura 67. Página Web, pestaña de gráficos de rentabilidad, Eurostoxx.

Aunque el segundo gráfico no se puede apreciar en la figura, estará colocado inmediatamente después del título que vemos al final de ella.

En este caso no es necesaria lógica adicional en java y simplemente usaremos etiquetas html con la ubicación de las imágenes en los ficheros.

Página Return VS Risk

En esta pestaña mostraremos según el mercado 6 gráficos, 3 en un euros y 3 en dólares. Cada uno de estos 3 gráficos corresponde a un periodo determinado 1, 3 y 5 últimos años. Para evitar tener 6 imágenes seguidas organizaremos por pestañas estás 3 imágenes, de manera similar a cómo representábamos las medidas de rendimiento, ver Figura 68.

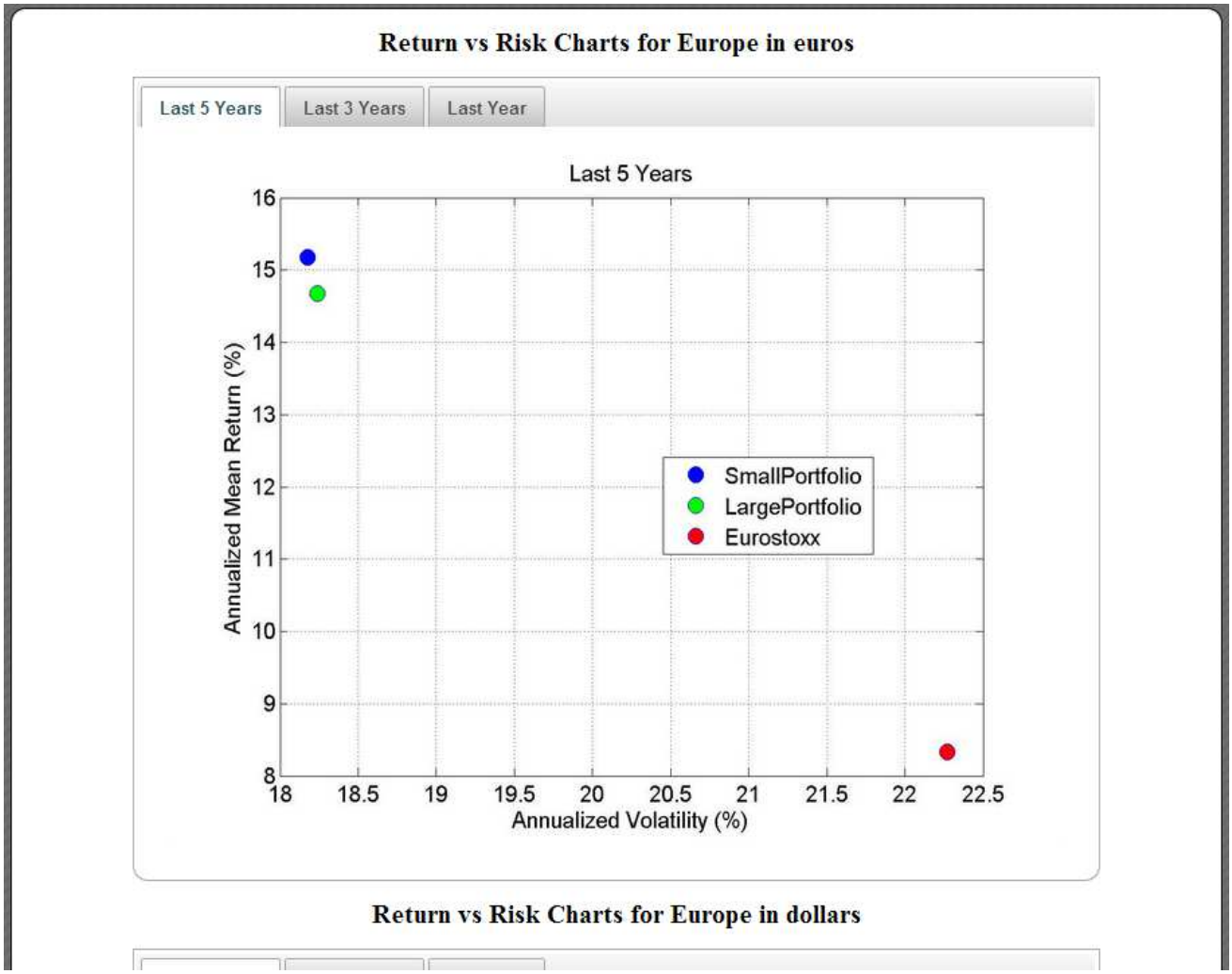

Figura 68. Página Web, pestaña de gráficos de Retorno VS Riesgo, Eurostoxx.

Los 2 grupos de 3 gráficos estarán ubicados uno después del otro. Al igual que pasaba en la página de Profits, no es necesario incluir lógica adicional en java para representar las imágenes, todo estará escrito en html con las ubicaciones de las imágenes. En ambas páginas (Profits y Return VS Risk) además mostramos primero la moneda nativa dependiendo del mercado elegido y después la alternativa. Las imágenes siempre estarán ubicadas en el mismo archivo con el mismo nombre, y será la rutina de actualización semanal la que se encargue de cambiarlas periódicamente.

Página Methodology

Esta página contendrá información sobre la metodología utilizada así como referencias a artículos y publicaciones relacionadas, ver Figura 69.

Quantitative Low-Volatility Investment

The investment recommendations are based on state-of-the-art statistical and optimization tools to select a portfolio with the lowest volatility. These portfolios are formed with the constituents of broad indexes, such as the S&P 500 and the EuroStoxx, and they are highly liquid portfolios that only take long positions.

Portfolios in US stocks are denominated in Dollars and also in Euros, which is useful for European investors interested in the US market. On the other hand, portfolios in Europe stocks are denominated in Euros and also in Dollars, which is useful for US investors interested in the European market.

Moreover, small and large portfolios are provided for each index, with around 10 and 25 stocks, respectively. Summarizing, the system recommends 8 portfolios every week: small and large portfolios for each index (S&P 500 and EuroStoxx) and for each currency (dollar and euro).

Although the portfolios are monitored on a weekly basis, we recommend rebalancing the recommended portfolio every two months.

Final warning: the recommended portfolio strategies are designed to have significantly less volatility than the associated market index. And they tend to capture the equity risk premium over longer time periods. But they may underperform the market index in sharply rising markets.

The methodology is based on the following publications:

Publications

- DeMiguel, V., A. Martin-Utrera, F. J. Nogales (2013). Size Matters: Calibrating Shrinkage Estimators for Portfolio Optimization. Journal of Banking and Finance, 2013.

Figura 69. Página Web, pestaña de metodología.

Sólo contendrá texto en html sin funcionalidad adicional.

Página About Us

En este apartado se realiza una breve presentación de los tutores y autor del proyecto. Con una imagen y una breve biografía, ver Figura 70.

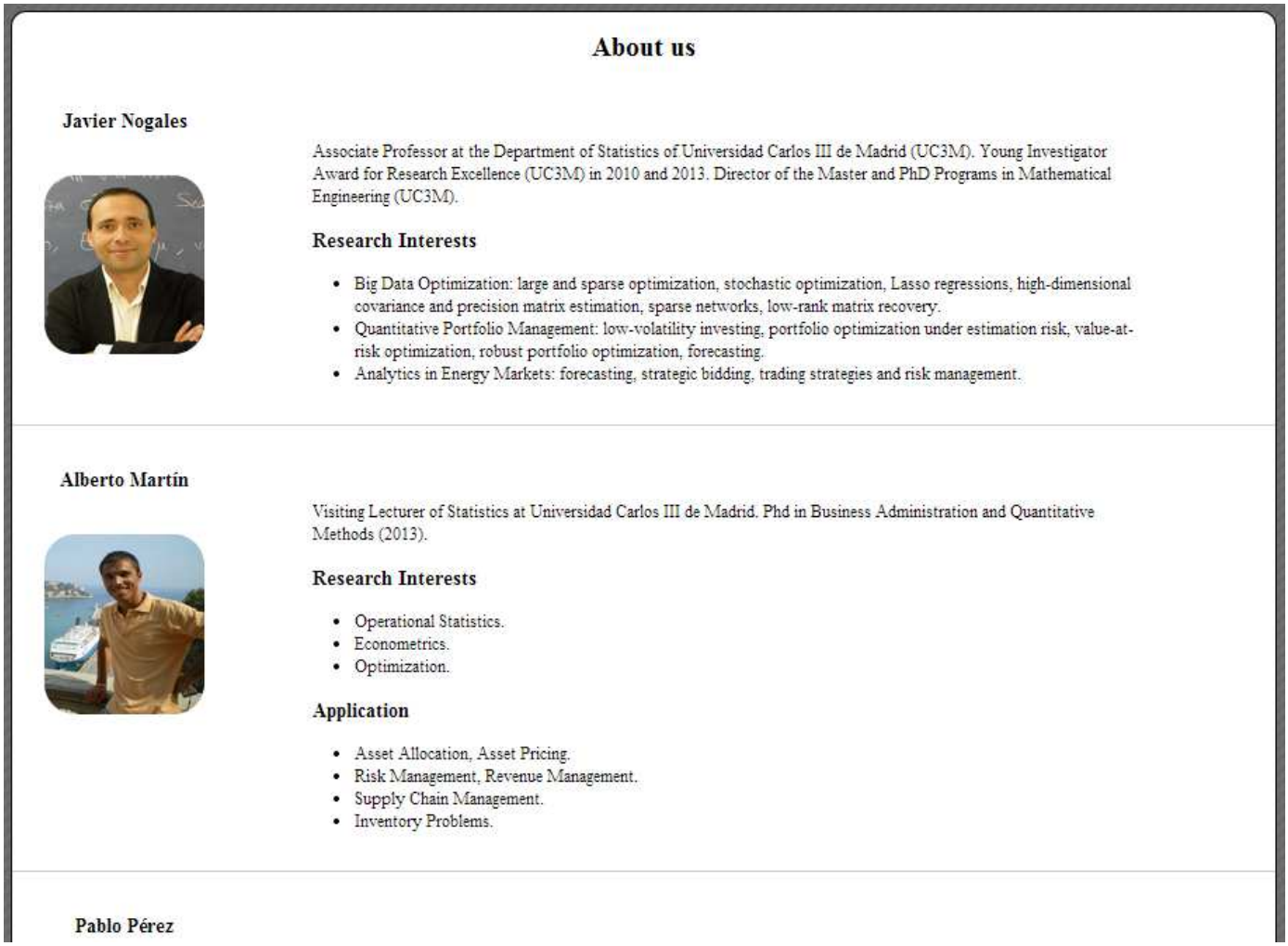

Figura 70. Página Web, pestaña sobre nosotros.

De nuevo la información mostrada será texto e imágenes estáticas sin lógica adicional.

Estructura Java

Se utilizarán principalmente 3 librerías externas al proyecto:

- Prime Faces 3.5.
- Java Server Faces 2.1.
- Java Csv.

Las primeras dos librerías están relacionadas, Prime Faces se basa en Java Server Faces para ofrecer una serie de funcionalidades y elementos optimizados que sean visualmente atractivos. Java Server Faces es también un *framework* que nos permite diferenciar fácilmente entre la parte visual y la programación de la lógica. La tercera librería nos facilita el manejo de archivos en formato *.csv* por lo que su uso afectará solamente a la parte funcional.

Cualquiera de las páginas que tienen lógica adicional o con datos dinámicos sigue la siguiente estructura ver Figura 71.

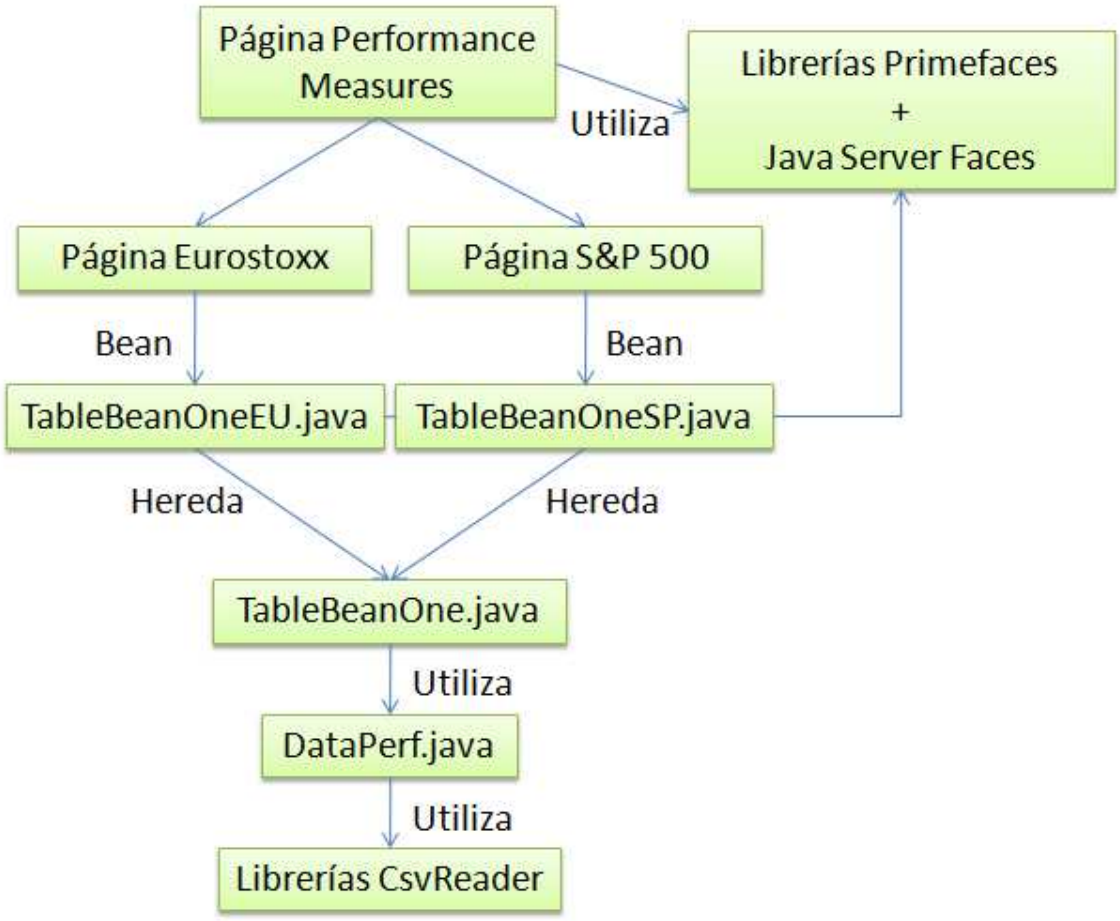

Figura 71. Estructura Web, y lógica adicional.

Las dos clases principales son *TableBeanOne.java* y *Portofolio.java*. En el ejemplo vemos cómo funciona la estructura con *TableBeanOne.java* para la página de Performance Measures. La clase *Portfolio.java* funciona de forma análoga con la misma estructura pero para la página de Portfolio.

Básicamente estas dos clases se encargan de gestionar los datos mostrados en las páginas de datos de rendimiento y la página de carteras. Cuando un usuario entre en una de estas páginas se ejecutará el método java constructor por defecto de estas clases y cargaremos dichos datos a partir de los ficheros de texto en arrays o listas de *Strings*.

Las tablas mostradas en las páginas *.xhtml* tomarán estos datos mediante una *Bean* ligada con los arrays de datos cargados en las clases java correspondientes, en el ejemplo *TableBeanOneEU.java* y *TableBeanOneSP.java* serán las clases que tienen definidas esta Bean ver Figura 72.

```
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;

@ManagedBean
@SessionScoped
public class TableBeanOneSP extends TableBeanOne{

   public static final String folderOneSP = "webInfoOneSP";
     public static final String[] currenciesSP = {"Dollars", "Euros"};
    public TableBeanOneSP(){
        super(TableBean.folder+TableBeanOneSP.folderOneSP+TableBean.sep,TableBeanOneSP.currenciesSP);
    }
}
```

Figura 72. Clase Java TablebeanOneSP.java.

Cómo vemos gracias a las librerías JSF es muy sencillo declarar una Bean a la que luego podremos acceder a sus atributos a través de las páginas web.

Finalmente en la página de carteras cuando un usuario selecciona una fecha y pulse el botón de enviar será también la clase *Portfolio.java* la que se encargue de comprobar, cual es la fecha seleccionada y cargar en los arrays correspondientes los datos de pesos y tickers de la nueva fecha. Cada vez que pulsemos ese botón ejecutaremos un método en la clase correspondiente que realizará estas actualizaciones. La primera vez que se cargue la página es cuando leemos los archivos y los almacenamos en variables java, por lo tanto cada vez que el usuario pulse la tecla de enviar (*submit*) no será necesario volver a leer los archivos y sólo será necesario acceder a la posición del array seleccionada y cargar los nuevos datos en la variable mostrada. De esta manera se evita cargar en exceso al servidor.

Capítulo 4

Presupuesto

En este capítulo presentaremos las fases de desarrollo por las que ha pasado este proyecto y se mostrará un desglose con los costes estimados de su implementación. Las fases aquí presentadas mostrarán el ciclo de trabajo de trabajo seguido durante la realización. Recordemos que todas las partes están relacionadas entre sí por lo que decisiones de diseño en la parte web pueden afectar a otras etapas anteriores. Se presentarán las fases siguiendo el orden lógico temporal que se ha seguido para la realización del proyecto. Empezando por las descarga de datos de servidores externos y terminando por la página web y el sistema completo.

4.1 Fases de desarrollo

A lo largo del proyecto podemos diferenciar varias fases, si bien hay que tener en cuenta que su ejecución no ha sido siempre de forma lineal debido a ciertas limitaciones y a que varios módulos del proyecto estaban ligados estrechamente, podemos enumerar las siguientes:

- Fase 1. Investigación y estudio de las herramientas.
- Fase 2. Rutinas Matlab.
- Fase 3. Página web.
- Fase 4. Sistema completo.

En función de las posibilidades que existían se iteró en varias ocasiones sobre estas cuatro fases optimizando y concretando diversas especificaciones.

Fase 1. Investigación y estudio de las herramientas

Durante esta primera fase se eligieron las herramientas que debíamos utilizar, desde Matlab, o el entorno de desarrollo web, al tipo de servidor o tecnologías de presentación de datos. Durante esta fase una de las fases más importante era buscar las fuentes de dónde obtendríamos los datos. Necesitábamos que el sistema fuese automático, por lo que la descarga de forma dinámica de datos era esencial. Se investigó de dónde obtener datos de cotización, cambio de moneda euro dólar y tickers de las empresas constituyentes de cada índice. Además se prestó especial atención al formato en que nos presentaban los datos para su posterior tratamiento.

En un principio se descargaban datos simultáneamente de Google y Yahoo para cotejar datos y solucionar errores, pero al extender nuestras carteras a Europa se eliminó la posibilidad de Google dado que no ofrecía datos para este mercado. Aunque esta parte también comprende el estudio de las tecnologías de la página web, temporalmente se realizó más adelante, justo antes de comenzar la fase 3. También se estudió cómo interactuarían todas las tecnologías involucradas.

Fase 2. Rutinas Matlab

Esta fase comprende toda la creación de las rutinas Matlab, desde la descarga de datos de los diferentes proveedores, hasta el volcado final de los resultados en ficheros. El trabajo realizado durante esta fase estuvo en constante cambio, los principales motivos fueron, la facilitación de acceso y presentación de datos al servidor, o la inclusión y eliminación de algunas especificaciones a lo largo del desarrollo del proyecto.

En esta fase también se incluyen todas las pruebas realizadas referentes tanto a la optimización de estrategias, cómo la elección de parámetros específicos o la optimización de algoritmos pesados en ejecución.

Fase 3. Página web

La tercera fase consiste en el desarrollo de la página web. En las primeras etapas del proyecto el servidor se ejecutaba localmente durante breves periodos de tiempo, es por ello que en esta etapa las ejecuciones de las rutinas de Matlab se realizaban manualmente. Para realizar esta parte era necesario tener los ficheros de datos generados por Matlab, por lo que a lo largo del desarrollo, si se decidían incluir nuevas medidas o gráficos esta parte se realizaba después que las modificaciones en las rutinas de Matlab.

Fase 4. Sistema completo

Una vez creadas unas rutinas estables y testeadas en Matlab y desarrollada una primera versión de la web se pasó a la implementación del servidor en una máquina remota. Al principio sólo se ofrecían datos estáticos y los cambios se

realizaban manualmente. Una vez fijadas todas las especificaciones de la página web se procedió con la implementación de las rutinas semanales y la actualización semana a semana. En esta parte se volvieron a retocar aspectos de las 3 fases, a la vez que se depuraban algunos errores que aparecieron tras las primeras semanas de su puesta en marcha.

Por último una vez el proyecto estaba en marcha funcionando de forma autónoma se redacto ésta memoria.
En la figura 73 se expone el diagrama de Gantt del proyecto.

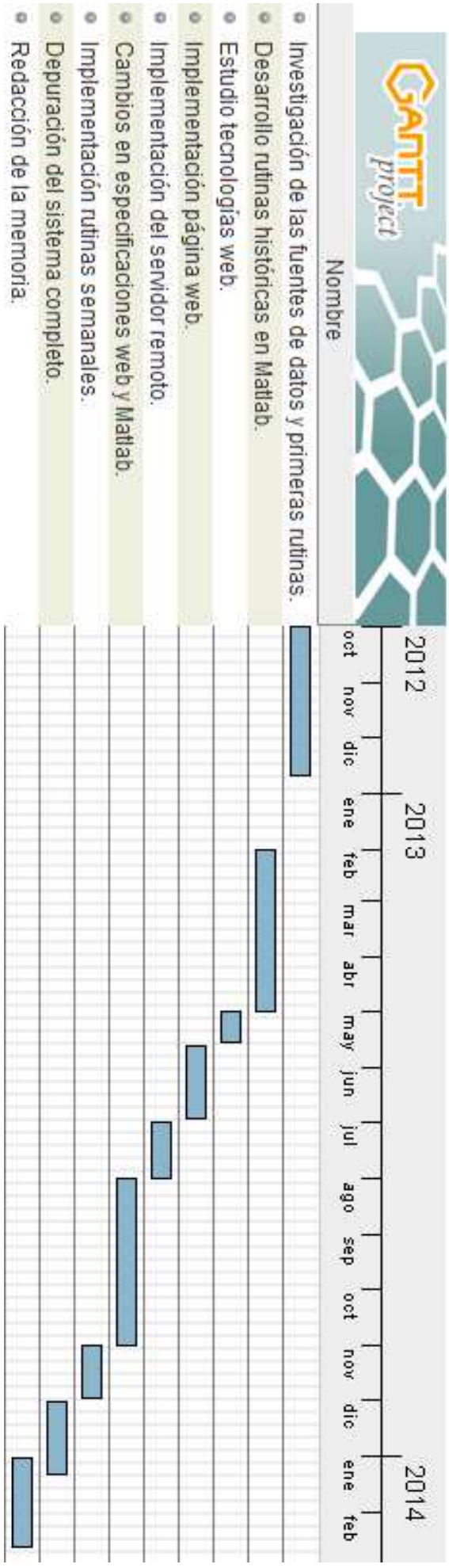

Figura 73. Diagrama de Gantt del proyecto.

4.2 Desglose de costes

Para realizar el presupuesto se tienen en cuenta tres grupos de coste distintos:

- Costes de personal involucrado.
- Material utilizado para la implementación y pruebas.
- Gastos directos del proyecto.

Además a lo largo de todo el cálculo la moneda utilizada será en euros, y los cálculos realizados se redondearán al segundo decimal.

Podemos ver el presupuesto completo en la Figura 74.

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:
Pablo Pérez González

2.- Departamento:
Estadística

3.- Descripción del Proyecto:

- Titulo: Infraestructura web para la implementación de estrategias de inversión automáticas de baja volatilidad.
- Duración (meses): 15
- Tasa de costes Indirectos: 20%

4.- Presupuesto total del Proyecto (valores en Euros):
39.989,02 Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)
Francisco Javier Nogales		Consultor Senior	1	4.289,54	4.289,54
Alberto Martín Utrera		Consultor Senior	2	4.289,54	8.579,08
Pablo Pérez		Ingeniero	7	2.694,39	18.860,73
		Hombres mes 10		Total	31.729,35

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Portátil	1.700,00	100	10	60	283,33
Servidor	1.500,00	100	7	60	175,00
Matlab Student Version	69,00	100	10	60	11,50
				Total	469,83

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
Internet	Movistar	600,00
Material de oficina		75,00
Electricidad		450,00
	Total	1.125,00

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,...

6.- Resumen de costes

Presupuesto Costes Totales	Presupuesto Costes Totales
Personal	31.729
Amortización	470
Costes de funcionamiento	1.125
Costes Indirectos	6.665
Total	39.989

Figura 74. Presupuesto del proyecto.

Capítulo 5

Conclusiones y líneas futuras

En este capítulo se explicará cuales han sido las conclusiones obtenidas con el desarrollo de este proyecto. Además se detallarán las posibles líneas futuras por dónde se podría continuar y ampliar el proyecto.
No sólo una vez puesta en marcha la plataforma sino también a lo largo del desarrollo se han podido extraer distintas conclusiones relevantes al proyecto. Se reflejarán además algunas de las dificultades experimentadas y se expondrán sus motivos.
Durante todo el periodo se han ido añadiendo y quitando funcionalidades, sin embargo se encontraron distintas características que podrían ser útiles para ampliar y continuar con este proyecto. Se expondrán las más importantes e inmediatas.

5.1 Conclusiones

El comportamiento de los mercados es siempre difícil de predecir, y lo que funciona en un periodo de tiempo puede no hacerlo en otro. Los mercados están siempre en continuo cambio y pasan por diferentes estados a lo largo de su vida. Las estrategias aquí implementadas intentan siempre minimizar la volatilidad de nuestra inversión.

Actualmente lo que ocurre en este sistema y podemos concluir es que las estrategias utilizadas consiguen su objetivo de minimizar la volatilidad, todas nuestras carteras a largo plazo obtienen mejores resultados que si invirtiéramos directamente en el índice en lo que a volatilidad se refiere. Además de reducir la volatilidad mantenemos unas rentabilidades comparables a las de los índices lo que puede hacer nuestro sistema más atractivo en momentos de incertidumbre o en inversiones a largo plazo.

Al inicio del proyecto al tener fijados 5 años de datos históricos el alcance de nuestras estrategias abarcaba gran parte de la crisis del 2007-2008 y nuestras estrategias superaban holgadamente a las rentabilidades obtenidas tanto en el Eurostoxx como en S&P 500, a medida que avanzaba el tiempo y se dejaba parte de la crisis fuera de nuestro alcance, ambos mercados experimentaron sobre todo el último año fuertes subidas y nuestras carteras perdieron parte de esa ventaja. Por lo que una desventaja de usar este sistema, es la pérdida de rentabilidad en periodos a corto plazo de fuertes subidas.

Para realización de este sistema ha sido necesaria la integración de varias tecnologías y disciplinas de diferente naturaleza. Por una parte son necesarios conocimientos básicos de estadística para poder calcular datos y optimizar las estrategias, todo ello realizado mediante el entorno Matlab. A partir de este punto es necesario tratar estos datos y presentarlos de forma atractiva, por un lado tendremos el servidor Tomcat y las páginas *.xhtml* con el que nos comunicaremos con el exterior, y por otro la lógica adicional en java para poder hacer dinámica la información mostrada. Todos estos elementos deben estar bien sincronizados para llegar al producto final.

El sistema propuesto en el presente proyecto llega a un amplio público con unas inversiones aptas para pequeños y grandes inversores que estén interesados en invertir en cualquiera de los mercados. Gracias al cambio de moneda ofrecido como elemento diferenciador de otros sistemas, puede resultar atractivo para inversores europeos que quieran invertir en el S&P 500 y usen su fondo en euros o para inversores americanos que deseen invertir en el Eurostoxx pero usen su fondo en dólares.

5.2 Líneas futuras

Las líneas de desarrollo en las que se puede ampliar este proyecto pueden ser de distinta índole. A continuación expondremos las más inmediatas, ya sean mejoras directas o ampliaciones de la funcionalidad.

- Ampliación de mercados. El sistema actualmente ofrece datos para Eurostoxx y S&P 500 pero sería fácilmente ampliable a otros mercados en ya sean empresas americanas y europeas o empresas de otros países. Rápidamente podemos ver que podría ser interesante ofrecer distintos cambios de moneda, de manera que expandiríamos el rango de clientes que podrían estar interesados.

- Extensión a mercados de renta fija. De igual modo que podemos expandirnos a otros mercados, también se podrían añadir otro tipo de activos como los emitidos a renta fija, también adecuados para las estrategias de baja volatilidad.
- Nuevas fuentes de datos. Los datos ofrecidos provienen de Yahoo Finance, sin embargo esto conlleva algunos problemas con la fiabilidad de los mismos, a lo largo del proyecto se han tenido que solucionar problemas con pequeños errores en los datos obtenidos, sería conveniente por tanto tener fuentes de datos alternativas. Estas fuentes permitirían cotejar los datos y completar aquellas partes vacías que a veces se encuentran en los datos de Yahoo así cómo detectar discrepancias entre ellos y poder ofrecer una solución. Podrían considerarse obtener los datos de alguna fuente de pago.
- Registro de clientes. Podría abrirse una nueva sección privada para usuarios que pudiesen cargar sus propias carteras en los mercados deseados y ofrecerles los mismos datos que ofrecemos con los resultados de sus carteras. También se podría ofrecer una comparación con nuestras estrategias.
- Sección de administrador. La inclusión de una sección enfocada a los administradores también sería interesante, con datos ampliados sobre el estado de salud de las carteras y comparaciones adicionales entre las tres estrategias disponibles. En esta sección también se podría permitir a los administrador cambiar parámetros de las inversiones, cómo el periodo de rebalanceo o la ventana de datos de entrenamiento.

Glosario

API	Application Programming Protocol
BPS	Basis Points
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
CVX	Disciplined Convex Programming
DNS	Domain Name System
HTML	Hyper-Text Markup Language
HTTP	Hyper-Text Transfer Protocol
IDE	Integrated Development Environment
IP	Internet Protocol
JSF	Java Server Faces
JSP	Java Server Pages
NaN	Not a Number
PHP	Hypertext Pre-processor
S&P	Standard & Poor's
SDK	Software Development Kit
SR	Sharpe Ratio
VaR	Value at Risk
XML	Extensible Markup Language

Referencias

[1] (06-02-2014) "*Servidores*" http://www.masadelante.com/faqs/servidor

[2] (08-02-2014) M. López "*Mercados de valores*" http://www.elblogsalmon.com/mercados-financieros/que-son-los-mercados-de-valores

[3] (07-02-2014) "*Índices de valores*" http://inverpedia.com/articulos/item/265-%C2%BFqu%C3%A9-es-un-%C3%ADndice-de-valores-y-cu%C3%A1les-son-los-m%C3%A1s-representativos?.html

[4] (09-02-2014) Funds Society "*Valores de baja volatilidad*" http://www.fundssociety.com/es/noticias/mercados/invertir-en-valores-de-baja-volatilidad-es-como-correr-el-tour-de-francia

[5] (09-02-2014) G. Neffa "*Volatilidad en los mercados*" http://www.saladeinversion.es/formacion/volatilidad-mercados-importancia-comprenderla/

[6] (15-01-2014) "*Matlab*". http://www.mathworks.es/products/matlab/index.html?s_tid=gn_loc_drop

[7] (15-01-2014) "*Matlab*" http://es.wikipedia.org/wiki/MATLAB

[8] (15-01-2014) S. Boyd and Lieven Vandenberghe "*Servicio Web*" http://es.wikipedia.org/wiki/Servicio_web

[9] (15-01-2014) "*Tomcat*" http://tomcat.apache.org/

[10] (15-01-2014) M. A. Alvarez "*HTML*" http://www.desarrolloweb.com/articulos/que-es-html.html

[11] (15-01-2014) "*Java*" http://www.java.com/

[12] (15-01-2014) "*Java Server Faces*" http://es.wikipedia.org/wiki/JavaServer_Faces

[13] (15-01-2014) "*Hojas de estilo CSS*" http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo
[14] (16-01-2014) "*NetBeans IDE*" http://es.wikipedia.org/wiki/NetBeans
[15] (15-01-2014) "*PrimeFaces*" http://www.primefaces.org
[16] (16-01-2014) "*Java CSV*" http://sourceforge.net/projects/javacsv/
[17] (15-01-2014) "*Convex Optimization*" http://stanford.edu/~boyd/cvxbook/
[18] (15-01-2014) "*CVX*". http://cvxr.com/cvx/
[19] (22-01-2014) "*Índice Euro Stoxx 50*" http://es.wikipedia.org/wiki/Dow_Jones_EURO_STOXX_50
[20] (22-01-2014) "*Índice S&P 500*" http://esbolsa.com/blog/bolsa-americana/que-es-el-sp-500/
[21] (27-01-2014) "*Basis Points*" http://www.investopedia.com/terms/b/basispoint.asp
[22] (27-01-2014) "*Ticker Symbol*" http://en.wikipedia.org/wiki/Ticker_symbol
[23] (27-01-2014) " *Índice Euro Stoxx 50*" http://www.wisegeek.com/what-is-the-dow-jones-euro-stoxx-50.htm
[24] (29-01-2014) "*Euro Foreign Exchange Rate*" https://research.stlouisfed.org/fred2/series/DEXUSEU/downloaddata
[25] (29-01-2014) DeMiguel, V., A. Martin-Utrera, F. J. Nogales (2013). "*Size Matters: Calibrating Shrinkage Estimators for Portfolio Optimization. Journal of Banking and Finance, 2013*."
[26] (29-01-2014) DeMiguel, V., L. Garlappi, F. J. Nogales, and R. Uppal (2009). "*A Generalized Approach to Portfolio Optimization: Improving Performance By Constraining Portfolio Norms. Management Science 55: 798–812.*"
[27] (29-01-2014) Jagannathan, R. and T. Ma (2003). "*Risk Reduction in Large Portfolios: Why Imposing the Wrong Constraints Helps. Journal of Finance 58(4): 1651-1684."*
[28] (29-01-2014) Ledoit, O. and M. Wolf (2004). "*A well-conditioned estimator for large-dimensional covariance matrices. Journal of Multivariate Analysis 88: 365–411*."
[29] (29-01-2014) "*Matlab Bug*" http://www.mathworks.com/matlabcentral/answers/83328-datevec-and-datenum-fail-intermittently-with-custom-format-string-matlab-r2010a
[30] (04-02-2014) "*Sharpe Ratio*" http://es.wikipedia.org/wiki/Ratio_de_Sharpe
[31] (04-02-2014) "*Value at Risk*" http://es.wikipedia.org/wiki/Valor_en_Riesgo
[32] (04-02-2014) Selfbank "*Maximum Drawdown*" http://www.rankia.com/blog/fondos-inversion-que-es/1137747-maximo-drawdown
[33] (05-02-2014) "*Beans*" http://es.wikipedia.org/wiki/Bean

[34] (27/01/2014) "*US Market Holidays*"
http://www.insider-monitor.com/market-holidays.html
[35] (27-01-2014) M. Krantz " *Índice S&P 500*"
http://www.usatoday.com/story/money/columnist/krantz/2013/06/13/sp-500-changes-index/2378221/